UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria

Corso di Laurea in Ingegneria Informatica

Tesi Di Laurea Magistrale

A Temporal Approach to Automatic Large Scale News Extraction

Laureando

Luigi Pio D'Onofrio

Matricola 462576

Relatore

Dott. Valter Crescenzi

Anno Accademico 2017/2018

Ringraziamenti

Grazie a tutti

Abstract

Nowadays news business has grown and more and more companies (e.g. Media Groups intelligence) make money out of news. The extraction of news is a key-feature for this business. This operation is difficult and usually hand-made for the most part. There is a need for automatic methodologies to recognize and extract news; This study aims to prove the validity of a temporal approach to assess the nature of a page guiding the extraction, along with a focused crawling solution for this domain.

To achieve this we collected sets of websites overtime and studied their behaviours, quantified these in certain features and measured how good was the information gain. We discovered that a temporal factor is always present in any news source, even if it is quite evident in some cases but less in others. Temporal information can be used to understand the nature of a page.

The results can be used both in the crawling phase, avoiding the crawl of useless data, and in the extraction phase: it is useful to know the class for a page because it enables the use of known extraction methods that need homogeneous data to work properly.

Introduction

Searching news has become "natural" for everyday life. More and more services and sites use news for several reasons, like news aggregations, news search services, business analysis, media Intelligence. All these services have their own mechanism to do news crawling and extraction based on some sort of heuristic or hand-made configuration files specific for every news source. This aspect should reflect the difficulty of automated news extraction. This is because the human intervention is still impactful, but also because the costs of this operation are really high, both in human employment and data amount terms.

These two aspects will guide this entire work. This work aims to provide a solution to the automatic news extraction problem along with the optimization of data usage both in bandwidth and storage.

There are a lot of related works in literature about the News Extraction problem because the topic has for sure a remarkable research value, but is really important also for the business. This is due to the fact that news (or data in general) extraction is fundamental in an era where Machine Learning and Big Data applications are such "hungry" of data. The current literature proposals or practical mechanisms to solve this problem rely on heuristics. These are in some cases template dependant, language dependant or massively based on the representation of the content.

The news domain is conceptually simple. It allows us to make assumptions due to this characteristic. In news domain we have a lot of elements, of few classes and all these elements have distinguishable behaviours.
We want to identify these behaviours and give a semantic to them to identify the correspondences `(page,class)`. To make this happen we need to observe the domain and his elements over time: this is the only way to catch evidences of some sort of dynamicity.

This work starts with an incremental problem definition and introduces some key concepts of this domain. After doing this it will continue giving an overview of the proposed temporal approach and how it is achieved.
A detailed explanation of the involved features and signals interpretations will follow, along with the needed concept and definitions for each one of them.
After this a multi-feature model, with its advantages and disadvantages will be proposed to the reader.

Given the reader a quick but clear overview about the domain, the problem and the involved key concepts, it continues with a definition of the final algorithm, used for the study of the approach but also being the basis for a production prototype of the system.
Here we'll discuss all the impactful aspects, along with a detailed analysis of the algorithm and its two phases. An interesting cost analysis is proposed right after the discussion about the algorithm. It compares the costs arising from a non-optimized use of data with the proposed solution. This aims to data usage optimization, both in bandwidth and in storage.

A simple reference architecture for the system will be provided next. After the reader is introduced to the algorithm, a quick overview of each component of the architecture will be provided to him, along with some considerations about scalability.

A quite substantial series of experiments was made during this study. The most relevant of them are provided to the reader and fully explained to better understand every choice taken during the entire study process. We'll start with a single-feature granularity heading toward a more precise one, like the proposed multi-feature model. The last series of experiments are made on full executions (simulations) of the algorithm in both its phases.

The entire idea behind this work was born and guided by literature works and state-of-the-art technical solutions. Many of these sources have been studied and analyzed and the most relevant ones are proposed to the reader, which will receive a brief explanation for each of them.

The last part of this entire work is dedicated to conclusions and future works. This work is a study and aims to validate an approach, but to produce a production prototype we need to analyze more aspects. Some of them are treated right in this section to let the reader imagine what this tool can be in the immediate future.

Contents

List of Figures

Chapter 1

Large scale news crawling

In this section the reader will be guided through a series of incremental problem definitions. This is due to let him better understand the problem and the domain we are dealing with.
He will be introduced to all the involved elements and their behaviours. These behaviours can be caught by some features which will be discussed in details further in this section, both individually and together.
In fact the last part of this section will provide an idea about a Multi-Feature model in which we'll explore advantages and disadvantages about the usage in conjunction of many features.

1.1 Problem Definition

Problem Definition 1. *Given one or more news source entry points, archive all the article pages from those.*

The presented definition is quite simple but we have to consider some aspects and challenges to conform our solution to the real world. One might be able to cover few *head* news sources with costly per-site and ad-hoc solutions, but there are a lot more *long-tail* sites that cannot be covered with bespoken solutions.

To construct these ad-hoc solutions costly human supervision is needed. Our problem definition needs to be complicated and substituted with a more precise one.

Problem Definition 2. *Given one or more news source entry points, archive all the article pages from these sources,* ***automatically and at scale****.*

"Automatically" means that we want to drastically reduce human supervision aiming to a totally unsupervised solution. That means that we want to involve humans only in a system parameters tweaking phase.
We target both head sources and long-tail sources with least human effort possible to reach a new level of coverage.

Another aspect we want to consider in the problem definition process is the data usage optimization. In this kind of study, time and spatial complexity are directly related. We can exponentially reduce time complexity reducing spatial complexity.
Data optimization is crucial to this kind of context because it can reduce noise into the data and lead to a more precise system along with remarkable economic savings. So our last problem definition is the following:

Problem Definition 3. *Given one or more news source entry points, archive all the article pages from these sources, automatically and at scale, optimizing data and bandwidth consumption.*

1.2 Actors involved in news sources

We are in a domain-based extraction field in which we identified two well-known main class of pages:

- **Articles**
- **Sections**

Definition 1. *Article*
*An **Article** is a page in which the main content is the news itself: the page exists just because the news itself exists.*

Definition 2. *Section*
*A **Section** is a page in which the main content is a collection of links to Articles: the page exists just due to the need of collecting certain news together.*

There is also another class, which is more precisely a meta-class, called **Other**.

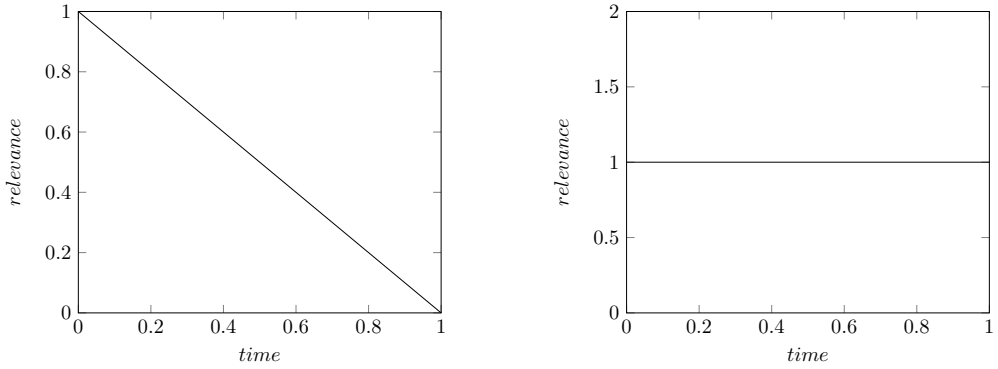
Definition 3. *Other*
*A page belonging to the **Other** class doesn't belong to any of the previously mentioned ones.*

Examples of these pages are contacts or the so called *"about us"* pages.
Each of these page classes have their own specific behaviours. Our goal is to identify them and give them a semantic. To achieve this we need temporal data to observe behaviours over time. This aspect defines the proposed temporal approach.

1.3 Temporal Approach

In this context we refer to a news website with *news source*. The main assumption of this work is that observing pages of a news source over time we can identify specific behaviours to classify each and every one of them. As said in the previous sections we have very few page classes. The first thing easily understandable is that an Article is an ephemeral (transient) concept: due to its nature we are not likely to see into a section page a year old news; this is because an Article lasts only for a certain period of time, which is generally short.

On the other end a Section is a durable concept; this means that a section page holds his relevance during time: A Section is rock steady, it is always visible to the user and it occupies always the same position in the page. It is a knowledge entry point, it needs to be easily reachable and the user doesn't want to search for it. We can refer to those aspects as **signals**.



(a) Articles relevance trend (b) Sections relevance trend

Figure 1.1: Relevance trend comparison between articles and sections

Definition 4. *Signal*

*A **signal** is everything that can give us a hint about the nature of a page.*

According to the aspects we just spoken of, we want to pick the signal about the *stability*

of a page.

1.3.1 Snap-shooting

As said in 1.3 we need to collect data over time. Let's give a definition of *snapshot*:

Definition 5 (Snapshot)**.** *A news source W crawled at a certain time t is a snapshot, and it is denoted with $S_{W,t}$.*

$S_{W,t}$ is an ensemble of pages of a website W crawled at a certain time t. After a specified period of time $t+1$ we re-crawl the website to collect $S_{W,(t+1)}$. The selected period of time between two snapshots should last enough to be able catch some evidences about pages behaviours. The choice of this period of time is crucial because:

- If it takes too long we could lose important information
- If it is too short we could waste resources or be led to biased information

This delay can be statically decided by the user when starting the system or it can be inferred dynamically during the crawling. We'll discuss this aspect later in this work.

1.4 Stability

Definition 6 (Stability)**.** ***Stability*** *is the degree of belief to find a certain URL at a certain depth in the next set of sample pages from a news source snapshot.*

Stability is a frequency and it is stated gathering URLs over-time. It has a value between 0 and 1:

- `0` means that we'll not find the URL in the next sample, for sure
- `1` means that we are sure we'll find the URL in the next sample

Figure 1.2: Stability Example

In 1.2 there is a simple example of what stability is: given a set of URL samples sets, stability gives us a frequency of crawling of an URL. The more a URL is crawled the more that URL is stable.

1.4.1 Strict definition

Definition 7 (Stability strict definition)**.** *Given a set of snapshots* $S = \{s_1, s_2, ..., s_n\}$*, with* $s_i = \{u_1, u2, ..., u_n\}$ *where* u_i *is an URL,* ***Stability*** *of an URL* u *is defined as*

$$Stability(u) = \frac{|S_u|}{c_s}$$

Where $S_u = \{s_i : u \in s_i\}$ *and* c_s *is the current snapshot.*

Basically we are calculating the mean of snapshots in which a certain URL appear over the entire sets of snapshots.

Note that we will refer without distinction to $S(u)$ and $S(p)$ where p is a page and not an URL: simply $S(p) = S(u_p)$ where u_p is the URL used to reach page p.

1.4.2 Signal interpretation

Stability is a tremendously simple signal to pick and it gives us a significant hint about the nature of a page. Referring to the previously treated classes, we could now distinguish between Articles and Sections. According to section 1.3, an **Article** is an

ephemeral concept so it is more inclined to vanish after a certain period of time and, on the other hand, a **Section** is more likely to stay still where it was placed. Both these aspects can be caught using Stability.

After a transitory regime, an **Article** will have a very low Stability percentage: as said, this is due to the transient nature of a news article, because it doesn't hold his relevance over time (1.1a).

Concerning **Sections**, they will have, after a transitory regime, a very high Stability percentage. This class of pages has a more stable nature, due to the fact they are made to index articles. Thanks to this characteristic the section holds his relevance over time (1.1b), it cannot simply disappear because it is like a lighthouse for the user. He needs sections to navigate the website.

1.4.3 Challenges and opportunities

As said, Stability is a very simple signal to understand and capture. Its main point of strength is that it represents a quite general aspect.
It doesn't rely on any technological heuristic like template-related or HTML-related ones. Simply looking at its value $S(u_p)$ we can state, after a transitory regime, the belonging class of a selected URL u of a page p: if $S(u_p)$ is high (let's say 1.0) then it is referring to a **Section**, to an **Article** otherwise.
The effectiveness of Stability, however, depends on some factors:

- **Global dynamicity of the website**: it is intended simply as its update frequency
  - It influences the time we need to reach the regime

- **Quite easily reachable Articles Cemeteries**: Defined in ??, they represents "dead" sections with no more updates
  - They contain dead articles too: if the dead section is easily reachable, those articles will be quite stable leading our system to a wrong classification
- **Crawling depth**
  - The more you go deep in the crawling the more we'll take to reach a regime
  - This happens because going deep we'll be more likely to encounter very old or dead pages

In real world examples, news sources are much more complex. Articles and Sections are not the only involved players. As said, there are *other* pages, which are uninteresting for the moment. These pages could have behaviors similar to Sections. Let's think to a privacy disclaimer page: it is always in the footer of the page, no matter what. This suggests us that Stability is not enough to distinguish between real Sections and Others.

1.5 Hypertextual Reference Dynamicity

To better understand the nature of a page we could exploit other aspects of the elements involved in this context. If we get closer to the page we can clearly see its content, which is, for the moment, intended only as its outgoing link occurrences. Link occurrences in a page have very distinguishable behaviours. We can exploit the dynamicity of these occurrences to give a semantic to our pages.

1.5.1 Links & Link Occurrences

Definition 8 (Link)**.** *A **link** l is intended as an hyperlink between two pages p_1 and p_2. it can be:*

- ***an outgoing link*** *for* p_1*: a link that starts from a page* p_1 *and leads to a page* p_2*.* $(p_1 \rightarrow p_2)$
- ***an ingoing link*** *for* p_1*: a link that starts from a page* p_2 *and leads to a page* p_1*.* $(p_1 \leftarrow p_2)$

Definition 9 (Link Occurrence)**.** *A* ***link occurrence*** o_l *is intended as an instance of a link* l *in a page* p*.*

A page p can have multiple link occurrences o_{l_i}. It is very important to distinguish between these two concepts. As we can see in 1.3, we have multiple link occurrences in the same page referring to the same link. In this case Section's links are both in the top navbar and in the footer.

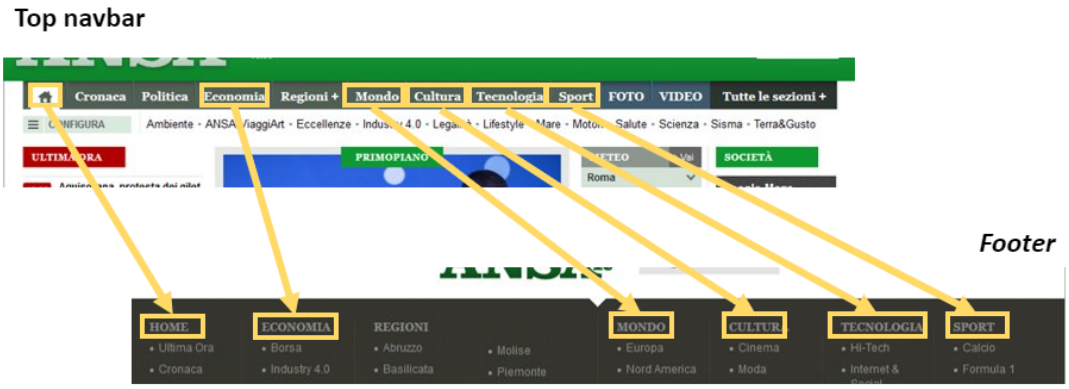


Figure 1.3: Example of multiple link occurrences both in navbar & footer

If we look at 1.4, we can see that after a small period of time the position of the highlighted link occurrence has shifted to a lower one and the last occurrence has *"disappeared"* to make place for the previous one. The zoomed part of 1.4 is a *Link Collection*.

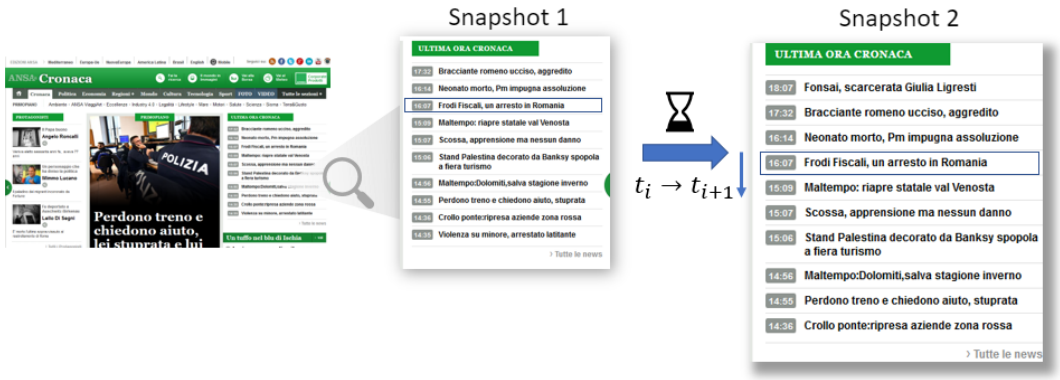


Figure 1.4: Example fo dynamic of a link occurrence

1.5.2 Link Collections

Definition 10 (Link Collections)**.** *A **Link Collection** is a particular group of link occurrences of outgoing links that are close to each other due to common characteristics (e.g. visual).*

The dynamic of a link occurrence o within his link collection may help to assess the nature of the target page.

Studying link occurrences only at a link collections level is quite important because this way we can be less affected by noises or errors but also to distinguish between two link occurrences of the same link.

1.5.3 Link Occurrences Behaviours

According to what was previously said, observing link occurrences inside link collection can help us state that, typically, link occurrences leading to Article pages are quite dynamic. These are characterized by frequent movement across all the pages. These movements could be time-based (the position changes over time due to the decreasing relevance of the article) or also voting-based (some part of the page can contain link occurrences leading to articles sorted by the amount of clicks or unique views).

Figure 1.5: Example of link collections. The yellow parts are the link collections while the purple ones are the link occurrences.

Basically, link occurrences, which are very dynamic in terms of XPath changes, will probably refer to an **Article** page. This is due to the transient nature of Articles and to the fact that their relevance has a monotone decreasing trend. This suggests us that Articles should not suddenly disappear to the reader eyes but they have to do it slowly, like a fading frame.

On the other hand, Sections link occurrences are, for the most part, always integral part of the template; they are always located into a template element like a navbar, a menu, or a footer. This behaviour indicates that links to Sections are quite static. This is due to the fact that the user needs to easily know where these "knowledge entry-points" are located.

So it is clear that watching at link occurrences behaviours we can understand the nature of the page leading that link. This is another interesting aspect to catch, let's call it *Hypertextual Reference Dynamicity.*

1.5.4 Definition

Definition 11 (Hypertextual Reference Dynamicity)**.** ***Hypertextual Reference Dynamicity (HRD)*** *of a link l is defined as the total amount of position changes, across adjacent snapshots, made by any link occurrence o of outgoing link l in every crawled page p.*

To be more specific **Hypertextual Reference Dynamicity** is defined as:

$$HRD(l) = \sum_{p \in P_l} \sum_{o \in O_{\to l}} M_p(o)$$

where:

- $P_l = \{p_1, p_2, ..., p_n\}$ are all the crawled pages in which there is at least one occurrence of outgoing link to l
- $O_{\to l} = \{o_1, o_2, ..., o_n\}$ are all the occurrences of outgoing link referring to l in the page p
- $M_p(o)$ quantifies the movement an occurrence o does in the page p

1.5.5 Conclusions

Hypertextual Reference Dynamicity can catch a quite strong signal, a real-world behavior. Hops of hypertextual references are something observable both from Humans and Machines. Usually the number of these hops can determine the nature of a page.

However it cannot still distinguish a **Section** from an **Other** page because their link occurrences are both steady.

1.6 Multi-Feature Model

At the moment we have two competitors: *Stability* and *Hypertextual References Dynamicity*. We could probably be wrongly led to choose only one of them. Stability is the most stable between the two and Hypertextual Reference Dynamicity is the fastest. In real-world examples there are some cases in which we can't simply choose the most stable or the fastest. This is due to the fact that not always we can afford waiting the regime reaching time of Stability, and not always we have a well-temporized website where we can perfectly catch the dynamic of a link.
There are cases in which Stability is the stronger signal and other cases in which Hypertextual Reference Dynamicity is the stronger one, while, there are some other cases in which combining the two leads to better results.

We thought of a **voting model** in which we can train a simple classifier (whatever method we want) on every combination of features, and let all of them vote (predict) the class of a page. The most voted class will be the predicted one.

This is a quite simple way to act and, due to the limited amount of features (and combinations), this approach is also quite fast (computationally speaking). We are not supposed to choose between features or doing a so-called feature selection, because we follow the principle that **the majority is right**.

Also the number of combinations without repetition is always an odd number:

$$\sum_{k=1}^{n} \frac{n!}{(n-k)!k!} = 2^n - 1$$

This means that we wont have any ties at all to care of.

1.6.1 Challenges

So, summarizing what we've just said, this model

- is quite fast due to the low amount of features
- avoids feature selection due to the simple principle that the majority is right
- has no ties to manage
- provides a performance boost in most cases
- protects the prediction to be affected by misleading features
- ensures a more stable classification

Both the involved features cannot still distinguish clearly a **Section** from an **Other** page. This is due to the fact that they share common characteristics both caught by these features. Let's take a Politics Section and a Privacy Disclaimer:

- They are both stable because both hold their relevance during time
  - Politics Section due to its knowledge entry point function
  - Privacy Disclaimer due to its legal relevance
- They are both steady because both have to be easy to find in the page

This is why we need to make a step further to refine the prediction.

Chapter 2

Algorithm

At the moment we are only able to distinguish between *articles* and *non-articles*(Sections and Others). This would be enough if our goal would have been to provide articles pages (of a specific news source) to a Wrapper[1] generator. This way we would guide the extraction of articles from a news source.
Providing homogeneous data to generate a wrapper to extract Articles is indeed part of the problem but it is not the whole problem. We need to be able to reach these articles. We need to know where to search for them, where to extract hyperlinks leading to them.

One approach, although inefficient, could be the one in which after generating the wrapper, we try to apply it to every single crawled page. Done this, we could evaluate the extraction with some sort of heuristics and accept only properly extracted data. This approach could be satisfying if we work only on a small set of news sources, but if we aim to scale on the size of this set, this one is not practicable.

[1]Wrapper in data mining is a program that extracts content of a particular information source and translates it into a relational form. Many web pages present structured data - telephone directories, product catalogs, etc. formatted for human browsing using HTML language. Structured data are typically descriptions of objects retrieved from underlying databases and displayed in Web pages following some fixed templates. Software systems using such resources must translate HTML content into a relational form. Wrappers are commonly used as such translators. Formally, a wrapper is a function from a page to the set of tuples it contains.[wik18]

2.1 Section-Driven Crawling

We aim to create a crawler which is able to acquire knowledge of the context it is operating in over time. Gathering information has not always the same duration. Some signals can be heard quite soon, some need more time. This said, we figured out that we could *exploit* signals in different times.

We theorized a 2-phased-crawler in which, for each of the two phases, we exploit different signals. After a certain period of time we think our knowledge is reliable enough to *exploit* it to do something finer. This first phase is called *Exploration Phase*. In this phase the system explores and observes the news source under examination to gather as much information as possible on the involved actors.

Once we reached the time we think our knowledge is reliable enough, another phase starts. This is called *Exploitation Phase*. In this phase we use previously obtained knowledge (exploit) to refine our understanding of the news source.

2.1.1 Preliminary considerations

Further in this section we'll discuss the two phases and the algorithm in detail. Before doing this we have to define and specify some parameters and terminology we are going to use from now on.

First of all we have to distinguish the two phases for the provided **entry points**. For entry points we mean the first page (or set of pages) our crawler is going to examine to create and explore the pages-tree. Making a distinction between different sets of entry points can help us to define the scope of our crawling in each phase.

Another quite important aspect to take in consideration is the **crawling depth**. Until

now crawling depth has always been fixed and chosen as a static parameter of the system. It was the same during the entire crawling phase. As said in 1.4 this aspect can affect classification (e.g. *Stability*) so it has to be wisely chosen for the specific task.

The last aspect we are going to consider is the **feature set**. It is interesting to reason about the features we are going to exploit in every phase and why. These are the key-points which will differentiate each phase.

2.2 Exploration

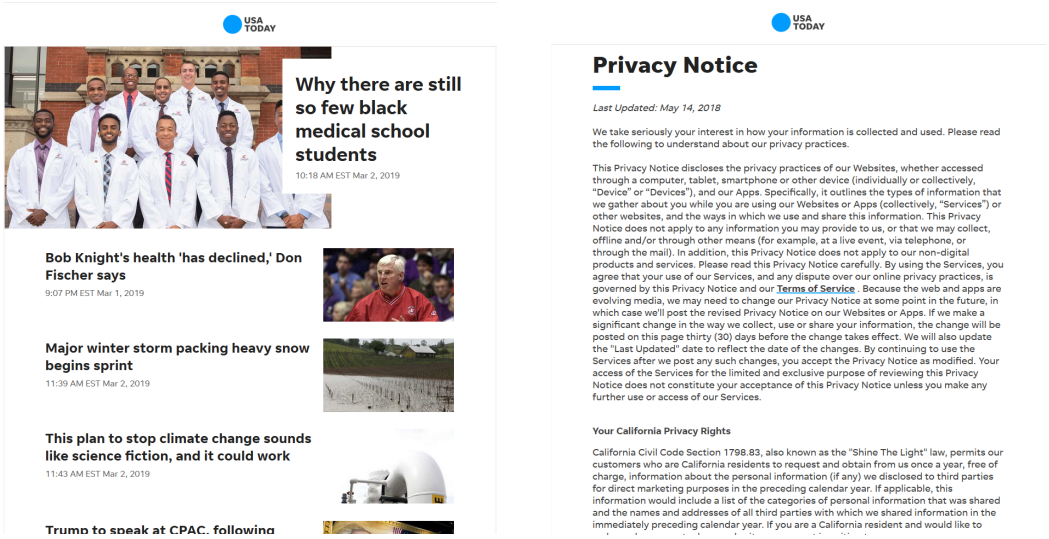
Exploration is the phase in which the system blindly explores the surroundings to gather knowledge. For our specific Section-Driven Crawler, this phase consists in the classification of pages in two simple page classes: *articles* and *non-articles*.
Articles are basically the pages belonging to the class defined in 1.2. Non-articles are pages not belonging to the previous class. We could watch at them as pages for which we are fairly sure they are not articles. They could belong either to Section or Other class.

This chapter started saying that understanding which pages are articles and which aren't is not the whole problem. That's true but it is for sure a starting point. Knowing this information with a good degree of certainty could guide the second phase of this methodology.
At this time it is clear we are going to exploit a preliminary, raw classification to refine our knowledge.

2.2.1 Sections & Others

As said in 1.6.1, Stability and Hypertextual References Dynamicity cannot distinguish between these two classes. This is because they share characteristics in these two dimensions.

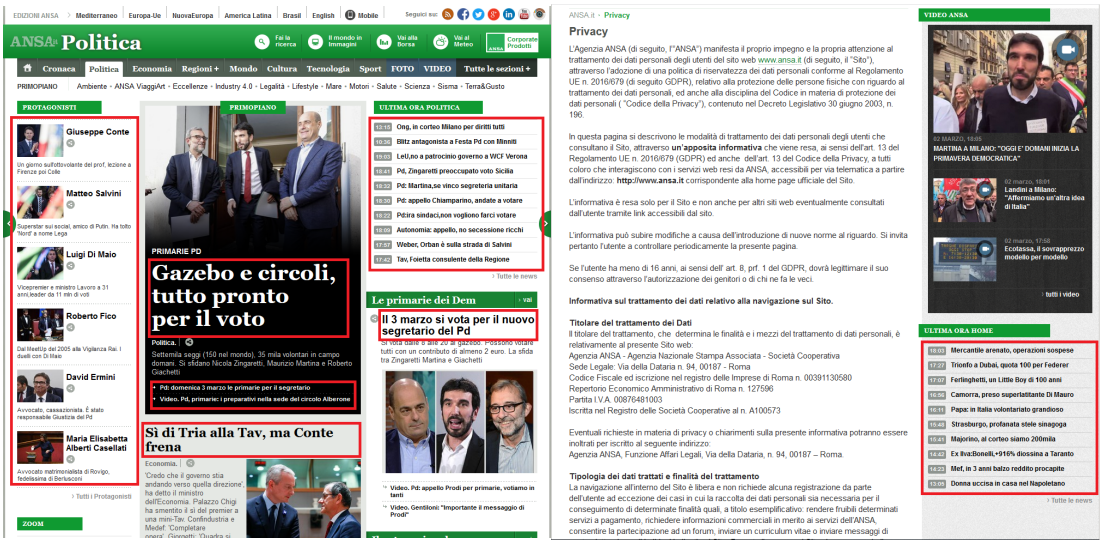
What is well known is that **Sections were born to point to Articles while Others are not**. So the amount of links pointing to articles can be a quite strong decider.



(a) A Section (b) A Privacy Notice

Figure 2.1: Two sample pages taken from *USA Today*

As we can see in 2.1a we have five visible hyperlinks to Articles where in 2.1b there is none. This is a quite strong signal to understand the nature of a page, so we could be tempted to classify as Other a stable page (high Stability an low Hypertextual References Dynamicity) without a single article link.



(a) A Section (b) A Privacy Notice

Figure 2.2: Two sample pages taken from *ANSA*

In 2.2b we can see we have a bunch of links to articles (along some links to videos, which are uninteresting for the moment). So, relying simply on the presence of links pointing to articles is not so good because the system would classify 2.2b as a Section. This said, is quite evident that, the amount of links to articles is instead quite different.

Watching just at the visible part we have 20 articles in the Section against the 10 of the privacy page. Also, we have to consider that the Section is still scrollable while the links to articles in the privacy notice are only the ones presented in the picture. It is clear that the amount of links is quite unbalanced and this can represent a quite strong signal to distinguish between Sections and Others.

2.2.2 Entry Points

In the exploration phase we can say that the target is the home page of the news source. This way we could see this phase as an *"assault"* to the entire website because we don't know how to navigate it yet.

Like in the real-world, if we want to read a specific article about politics from *GenericNewsSource.com*, we visit the home page, because we don't know the URL for the section. Found that, we can easily reach the Politics section and search for the article we are interested in.

The home page is the right place to start gathering information when you don't know anything at all about that news source.

The home page contains

- hyperlinks to all the main sections
- some links to articles belonging to different sections (the most relevant ones)
- the majority (if not all) the pages which are neither articles nor sections

In this phase we are crawling the news source completely blindly. This motivates the need of a generic information source, like the home page. At this point, and during the entire Exploration phase, the system doesn't know anything about the news source it is operating on.

2.2.3 Crawling Depth

As said in 1.4, performance is affected by crawling depth, so choosing the right value for this parameter could be crucial. What we discovered, during the study of this approach, is that a low depth (e.g. $depth = 1$) corresponds to a quite good performance when it

comes to articles vs. non-articles classification.
This is due to the fact that Stability takes less time to converge (1.4) and more in general, the data we are going to train the system is way less noisy.

So for the Exploration phase we can afford a low crawling depth because it is enough to gather all the necessary to provide a good classification. In addition to this, we can also reach the phase-switch faster.

2.2.3.1 Low Depth Crawling Considerations

To reach the phase-switch with a reliable knowledge base we need to explore at least the main sections of the website. These sections are quite easy to reach from the home page (this is one of their main charcteristics). So at a $depth = 1$, we'll actually visit these pages and we have their content under our lenses.

It is interesting to notice that to evaluate Stability and Hypertextual References Dynamicity of a link we don't necessary need to visit the page; we only need to spot at least one link occurrence leading to this page.

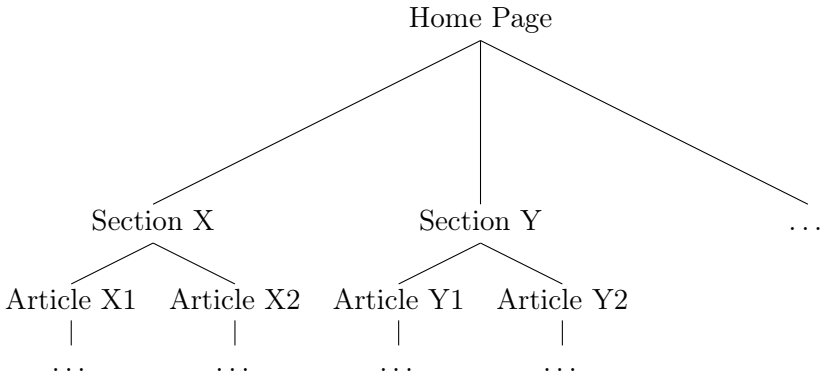


Figure 2.3: Example of news source tree

As we can see in 2.3, the home page is positioned at $depth = 0$, while sections are reachable at $depth = 1$. We don't need actually to visit Article X1 page at $depth = 2$

to quantify Stability and Hypertextual Reference Dynamicity for it. That's because we can observe link occurrences for Article X1 (and Article X2 aswell) from Section X page (for Section Y is the same).

More in general we can say that if $depth = d$ we can observe the behaviours of all the pages lying at $depth = d + 1$, because we can already observe them from 1-level higher perspective. As we'll discuss later, this is a double win both for performance and data optimization.

2.2.4 Features Set

The feature set we are going to use in this phase is basically made by Stability and Hypertextual References Dynamicity. These are the most impactful features when it comes to articles vs. non-articles classification. They are both quite strong signals representing real-world phenomena. This is their key-aspect and this guarantees a good behaviours extraction, for sure.

The process could be faster for some news sources or slower for others, but the result is guaranteed in almost every scenario.

2.3 Exploitation

Once the system has a reliable knowledge base, it can now exploit what it learnt during this time.
As we said in 2.2.1, these two classes are well distinguished by the amount of links pointing to articles. We can imagine that at this time we have a knowledge base made of tuples `(page, temporary-class)`, like

section.html	non-article
article1.html	article
privacy-disclaimer.html	non-article
article2.html	article
...	...

Table 2.1: Knowledge-base example at the end of exploration phase

When taking snapshots the system stores also the parent of a page, so we can easily group this by their parent page. This way we are building a so-called *Typed HyperGraph*. This structure is a graph, in which nodes are the pages, along with the predicted type, and the arches are *father-son* relationships.

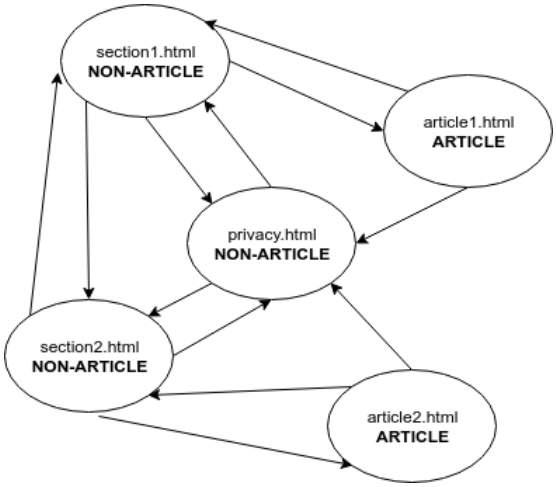


Figure 2.4: An example of a Typed HyperGraph

Using this structure we can now spot the real nature of those elements labeled as Non-articles. To do so we only need to count how many outgoing arches point to articles for each node. Those with an high amount of articles will be labeled as *sections*, otherwise they are labeled as *others*, as we can see in Figure 2.5.

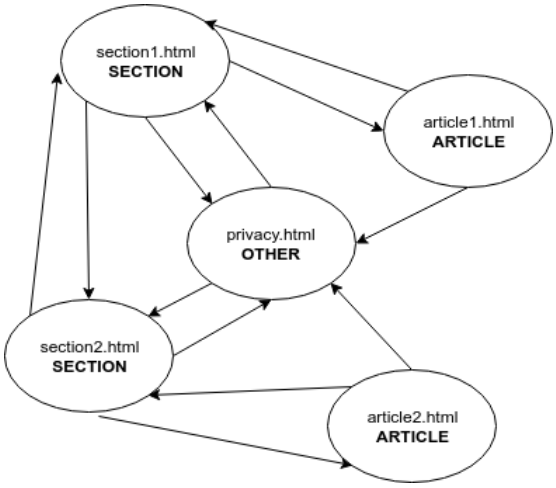


Figure 2.5: An example of a Typed HyperGraph after exploiting knowledge

From now on our system will be aware of all the three page classes involved in this domain: Articles, Sections and Others. Iteration after iteration it will refine its knowledge base along with the Typed HyperGraph to be the most accurate possible.

2.3.1 Entry Points

In this case the scope of the crawling phase changes drastically. The system doesn't do "assault" anymore to the entire website, but, exploitating the just obtained knowledge, which is refined iteration after iteration, it can now target only the knowledge entry points. In other words only the sections. Doing this, we can target directly the zones in which the information we need lies. At this moment the system knows exactly where to find information and what kind of information it expects to find. This way the system can avoid noisy and useless data.

2.3.2 Crawling Depth

Speaking of crawling depth, from now on we can "virtually" increase the depth we are searching information at. As said the new entry points are located at a greater depth than the starting one (home page). But it is not just that, basically we are avoiding

the visit of level 1 useless links like articles or other, which could be redundant for a lot of snapshots. So we are virtually visiting depth 2 but the entry points amount is lower than the total amount of level 1 links. This could increase performance but also optimizes data usage.

2.3.3 Features Set

In this phase the features-set is expanded with a couple new features called *Links To Articles Count* and *Hypertextual Content Dynamicity.*

2.3.3.1 Link to Articles Count

This is a very simple feature at this point. Let's give it a definition

Definition 12 (Links to Article Count (L2AC))**.** *Given a page p, Links to Articles Count is defined as the amount of links pointing to articles in the page.*

As we previously said, this is very useful to make distinctions between Sections and Others. This is the its main function. It needs previous knowledge to be built overtime and it is less affected by errors because we exploit it only when we are enough sure our knowledge base is reliable.

2.3.3.2 Hypertextual Content Dynamicity

Definition 13 (Hypertextual Content Dynamicity)**.** *Given a page p, Hypertextual Content Dynamicity quantifies the dynamicity of the outgoing link occurrences from p.*

This is a feature exploitable from any point and any phase. It is also used to distinguish between Sections and Others but it is less precise because it can be affected by some noise in the data. We could use it in the Exploration phase to reduce the amount of false positives and false negatives. Moreover in the Exploitation Phase it is used to reinforce the prediction made adding L2AC to the features set. However we could think

that it as an optional feature.

To provide a more precise definition, we can say

$$HCD(l) = \sum_{p^* \in L_{\to p}} HRD(p^*)$$

where $L_{\to p} = \{p_1^*, p_2^*, ..., p_n^*\}$ such that $\forall {p^*}_i \exists p \to {p^*}_i$
The basic idea is:

- in **Sections** we have a lot of articles that are pointing link occurrences → their hypertextual content is very dynamic due to their high HRD.
- in **Articles** we have, in the worst case scenario, a bunch of articles-pointing link occurrences, the rest are only sections/others-pointing link occurrences → their hypertextual content is quite static due to the low HRD value of all these elements
- In **Others** we should have a behaviour very similar to the one registered with articles

2.4 The algorithm

In this section we'll provide the pseudo-code of the proposed solution. We'll also provide a brief description of all the involved steps.

Algorithm: Section Driven Crawling Algorithm
Data: One or more home pages as entry points
$entrypoints = \{h_1, h_2, ..., h_n\}$;
$depth = 1$;
$features = \{S, HRD\}$;
while *not phase-switch condition* **do**
    /* EXPLORATION */
    foreach h_i *in entry-points* **do**
        Crawl h_i with provided depth;
        Extract features from crawled data of h_i;
        Train a classifier (articles vs. non-articles) using extracted features;
        Evaluate model updating data for the phase-switch;
        Update sampling frequency for h_i;
        Update crawling depth for h_i (optional);
    end
end
build the Typed HyperGraph;
$features = \{S, HRD, L2AC\}$;
update entry points including sections-only pages;
update depth (optional);
while *not termination condition* **do**
    /* EXPLOITATION */
    foreach s_i *in entry points* **do**
        Crawl s_i with provided depth;
        Extract features from crawled data of s_i;
        Train a classifier (articles vs. sections vs. others) using extracted features;
        Update sampling frequency for s_i;
        Update crawling depth for s_i (optional);
        Update knowledge and Typed HyperGraph;
        Update entry points adding new sections and removing false positives;
    end
end

Figure 2.6: Section Driven Crawling Algorithm

The algorithm under examination is quite simple. In each phase the system does a cycle:

1. Crawling
2. Features-Extraction
3. Classification
4. Model Evaluation
5. Parameters Update

Each of these steps is guided by the parameters calculated at that specified time. In the Exploration phase we **crawl** the news sources starting from the home page, at $depth = 1$.

When the system reaches the **features-extraction** phase, it will extract Stability and Hypertextual References only. This is due to the fact that these two features are the right ones that catch evidences usefully to make distinctions between articles and non-articles. They are also quite effective with at low depth of crawling.

Features-Extracted data will feed the **Classification** phase in which the system builds a model to make distinctions between the two classes. At the moment it is possible to choose which algorithm the user wants. This aspect wil be however treate further in this work.
Model Evaluation is a bit tricky because it depends on the model itself and it will also be treated further in this work.
The last step of the cycle is instead the **parameters update phase**. In this phase parameters such sampling frequency are updated to optimize data utilization and reduce the bias. The optimization is heavily based on the update frequency of the website which is, as we'll see later, understandable using different factors.

Once a phase-switch condition is reached, our system will update the knowledge-base and will build the Typed HyperGraph which will be exploited both in the crawling step, due to the enriched entry points set, and in the features-extraction step in which we'll introduce Links to Articles Count as a feature.
In this moment we could also update the depth we are going to execute the crawling at. This would be more appropriate for quite small news sources. However it is completely optional.
From now on the system is aware of the news source it is operating on and, step by step, will refine its knowledge, leading to a way more precise crawling and optimized data usage.

2.4.1 Classification considerations

For this kind of work we used a very simple supervised solution to the classification task. We chose to use decision trees. This is because they are a very simple structure which allowed us to control them easily. They are also very fast to train, which is quite useful during the study of an approach. In this case, in fact, it is easy to defocus from the target if the time you need to test a feature is too long.

As described in section 1.6, to ensure the correctness of the prediction, we use a voting model.
In our case we trained a decision tree for each combination of features we have in the features set.
This way we can both avoid performance to be affected by misleading features and boost them in the best case scenario.

The system is made in a way that it is possible to provide any kind of classification

algorithm. The only constraint is that the chosen algorithm needs to have some features in input and has to provide a model capable of predicting the page classes, exploiting these features.

This means it is possible to use a supervised approach (as the one of this work), unsupervised, or a custom one. Another constraint is represented by the voting model because this way we can also avoid features selection, which could affect the usability of the system.

In the last chapter of this work we'll also provide some ideas to implement an unsupervised solution, which will guide this tool towards a more automatic trim.

2.4.2 Model Evaluation & Phase-Switch condition

As we said in the previous section, Model Evaluation is quite tricky because it massively depends on the chosen classification type.

It is known that the two main families of classification algorithms are **supervised algorithms** and **unsupervised algorithms**.

For the validation of this approach we obviously used a supervised one (as previously said). This involves the presence of ground truth data, which is both used during classification and evaluation. With such data it is very easy to evaluate the rightness of a solution, it is just a number. If it is possible to calculate this number the evaluation is very simple and this is the way we studied the temporal approach.

However, creating that ground truth data is an hand-made work. So this is not automated at all. It is only a good approach to validate a solution, nothing more.

This work aims to provide a totally automated solution. In this case unsupervised approaches are necessary, but without ground truth data is not so easy to evaluate a model anymore. We identified some indexes that don't need ground truth data but are

simply evaluations of the characteristics of the sets of classified data.

The **Entropy**[2] **Index** is one of them. It can quantify how much are differentiated the elements of a specified set.
We could follow the principle for which a good classification is that one in which the classified (clustered) sets of elements have a very low entropy.

Another index could be the one called **Silouhette Index**. This refers to a method of interpretation and validation of consistency within clusters of data. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).
The silhouette ranges from -1 to $+1$, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters.
The silhouette can be calculated with any distance metric, such as the Euclidean distance or the Manhattan distance[wik19d].

Using these two indexes, for example, it is possible to define a threshold. After this threshold we could consider that the classification made by the system is reliable enough to switch to Exploitation phase.

[2]Lack of order or predictability; gradual decline into disorder.

2.5 Challenges and Opportunities

In this section we are going to examine all the opportunities and challenges offered by the presented solution. Section Driven Crawling main objective is to boost classification accuracy, leading to the classification in the three main classes of interest: Section, Article and Other. This result would be more difficult to achieve without exploiting some knowledge at a certain time.
It also provides lower crawling depths in general which led to better performance in Stability and Hypertextual References Dynamicity (as we can observe in the Experiments & Results chapter). In addition it causes a massive reduction in downloaded and stored data.

Data optimization is crucial for this context, a single level of depth, multiplied for many news sources could mean terabyte of useless data.

2.5.1 Cost Analysis: Amazon S3 Storage

In this subsection we'll try to make some estimation of the amount of money a company could save using this approach. Let's take in consideration a common Cloud Storage service like the well-known Amazon S3.

Pricing	
S3 Standard Storage	
First 50 TB / Month	$0.024 per GB
Next 450 TB / Month	$0.023 per GB
Over 500 TB / Month	$0.022 per GB

Figure 2.7: Amazon S3 pricings for storage service

In Figure 2.7 we can see the monthly price per GB Amazon offers to its customers. Amazon also provides a useful website to do a fast calculation of the monthly cost for its service (https://calculator.s3.amazonaws.com). All the numbers we'll see further in this sections have been calculated using this tool.

Imagine crawling a website

- 100 snapshots
- every page is 1MB
- every database row for a link occurrence is 5KB
- 100 link occurrences per page
- 50 crawling frontier links per page
- store data on *Amazon S3 - Standard Storage*

2.5.1.1 Generic Crawler

Characteristics:

- $depth = 2$
- 100 snapshots

To crawl a single snapshot we would need **3,82 GB** of data. This is due to the fact that at

- **Depth 0**
  - 1MB + 100 link occurrences ($100 * 5KB = 500KB$)
  - **Let's simplify: 1.5MB per page**
- **Depth 1**
  - 50 pages = **75MB**
- **Depth 2**
  - 2500 pages = **3750MB = 3,75 GB**

So, the amount of data needed for 100 snapshots, using a generic crawler, would be **382,65 GB**.

2.5.1.2 Section Driven Crawler

Characteristics:

- 2 phases: *Exploration, Exploitation*
- $depth = 2$
- 100 snapshots

Exploration

- **Depth 0**
  - 1MB + 100 link occurrences ($100 * 5KB = 500KB$)
  - **Let's simplify: 1.5MB per page**
- **Depth 1**
  - 50 pages = **75MB**
  - If this phase lasts for all the 100 snapshots, we'll have 7650MB (7,65GB) (worst case scenario)

Exploitation

- Let's say Exploration phase lasts on average 50 snapshots $\rightarrow 76,5 * 50 =$ **3,825GB**
- **Depth 0**
  - **sections are entry points (virtually is a depth 1)**
  - 20 pages = $20 * 1.5 =$ **30MB**
- **Depth 1**

- 1000 pages ($1000 * 1.5MB =$ **1,5GB**)
- For the remaining 50 snapshots $= 1,5GB * 50 =$ **75GB**

So, the amount of data needed for 100 snapshots, using a Section Driven Crawler is 3,825GB (50 snapshots Exploration) $+ = 1,5GB * 50 =$ **75GB** (50 snapshots of Exploitation) $= 80GB$

2.5.1.3 Conclusions

If we use the previously provided link to make an estimation of the monthly cost, **using only storage as a parameter** (it is possible to specify more!)

Without any optimization

- 100 snapshot - 1 news source = 382,65GB
- 100 snapshot - 1000 news sources = **382,65TB**
- Monthly cost for 1000 news sources = **9969.59$**

With Section Driven Crawling

- 100 snapshot - 1 news source = 80GB
- 100 snapshot - 1000 news sources = **80TB**
- Monthly cost for 1000 news sources = **737.16$**

This result means that we could save up to 93% in money, maintaining the same, or get even better, performance. This is a very good result, even if it is only an estimation. If we want to scale on an high amount of news sources optimization is crucial to not waste useless resources.

Chapter 3

A prototypical architecture

In this chapter will provide the reader an overview of a general architecture (at high level) of the entire system. The design of this architecture is guided by the architectural pattern called **Pipes & Filters** (with some differences).
The elements of this architecture are quite few and with well-defined responsibilities. The final result is computed by a series of iterations for each of the two phases, as we've told in chapter 2.

We will provide a quick overview for each of the elements along with their relationships. We'll discuss some scalability aspects in addition to a simple definition of the involved API interfaces.

3.1 Pipes & Filters

Pipes & Filters is made up of two types of elements:

- Filters
- Pipes

The first category of elements is made of components that transform (or *filter*) data, before passing it, via connectors (pipes) to other components. The architecture is often used as a simple sequence, but it may also be used for very complex structures.

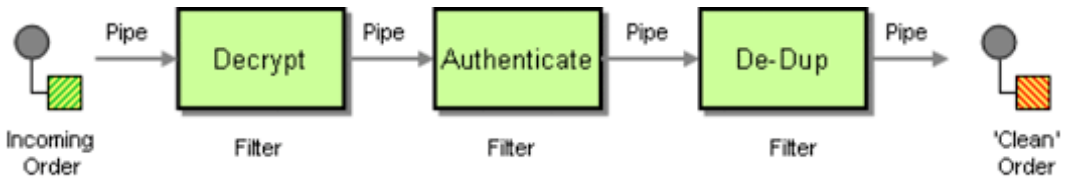


Figure 3.1: an example of Pipes & Filters

In Figure 3.1 we can identify green components as filters, because they transform or filter the data they receive via the pipes connected to them. A filter can have any number of input pipes and any number of output pipes.
The pipes are the connectors that pass data from one filter to the next. Obviously there is always a data source and someone interested in receiving the final result of the processing.

3.1.1 Considerations

This architectural pattern is simple but is satisfies many quality requirements of our system. The most relevant ones are **Modifiablity** and **Performance**.
Being able to dynamically insert, remove or swap filters (Modifiablity) is a key feature for a system like the proposed one. The main reason is that this way we can more efficiently study new methodologies and approaches by just adding or removing filters.

On the other hand the system has some filters which need the same data but in different formats, resulting in different pre-processing. So instead of duplicating filters, including the specific pre-processing one, we can model this atomic step as a distinct filter enabling us to combine them to obtain specific behaviours.

This architecture also guarantees quite good performance horizons. Filters are good concurrent units, they can both run serially or run in parallel for maximum performance. It is also possible to run many instances of the same filter to achieve load balancing reducing the risk of bottlenecks in the system.

Theoretically, Pipes & Filters is also good for **Availability**. At the moment this is not a crucial aspect for our system, due to the fact that it is just a prototype. But, when it comes to production environments, availability could be relevant for the business and loosing this quality could result in loosing money.

3.2 Components

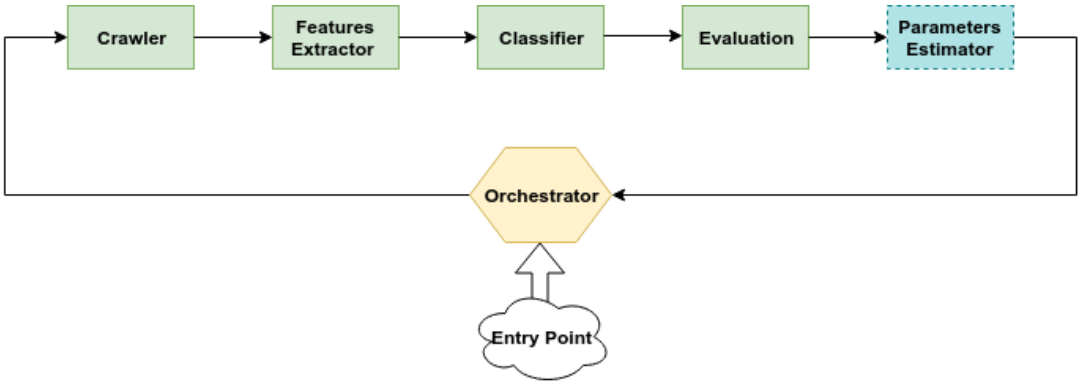


Figure 3.2: System Architecture

As we can see from Figure 3.2, the computation is cyclical, so source and destination collide. At every iteration we have a series of steps processed by filters (the ones in green, the blue one is optional).

3.2.1 Orchestrator

This component is not actually a filter but it simply manages a cycle for each of the provided entry points. It receives requests to start a processing task and its interface only needs an entry point (an URL) and optionally the amount of time (or snapshots) the system have to run on that entry point.

It is a stateful component because it needs to store data about each of the entry points until the cycle ends. We can still scale horizontally, replicating this element, but we still need a mechanism to select one.

It also receives queries from other components to provide them some key information.

3.2.2 Crawler

It offers the crawling functionality. This component executes the crawling at the provided depth, starting from the also provided entry point. This two parameters constitute its interface.

Crawling is a very expensive task and due to this characteristics it is clearly oriented to being distributed. We kept this in mind and modeled this component to be as much stateless as possible. This way we can optimize resources and parallelize executions to maximize performance. Crawled data is then optionally stored someway before passing data to the pipe leading to the next filter.

This could be achieved both using local storage or a Cloud Storage service.

3.2.3 Features Extractor

Data coming from Crawling ends in many Feature Extractors, each one extracting a specific feature. Each Feature Extractor is chosen dynamically, depending on the phase we are in, querying the orchestrator.

All these features are computed by consecutive aggregations, to maximize performances. Once the current computation ends it aggregates the results to the previous one and starts up the second feature extraction and so on. Once all the features are extracted, data is sent to the next filter.

This can be seen like an internal cycle in which, for each features in the set, the system executes the extraction. Once the cycle ends data is sent to the next filter.

This is not actually optimized because this behaviour could be obtained also combining pipes & filters but it would give rise to the problem of selecting a specified pipe or a specified filter to re-compose all the extracted features for a specific source or entry point.

3.2.4 Classifier

This components takes features-extracted data and simply trains a classifier on it, following the constraints provided in subsection 2.4.1. Once the model has been built, classes are predicted and dynamically the Typed HyperGraph will be populated or refined and then stored.

The output model is trained on features-extracted data relative to the last snapshot. As previously said the classification algorithm can be whatever the user wants, the only requirement is the voting system, needed to make the prediction stronger.

It is completely stateless and we can replicate it to scale horizontally. The result of this phase is sent to the next filter which will evaluate its goodness.

3.2.5 Model Evaluator

This components is responsible in giving a score to a model. The evaluation can be general or algorithm-specific. In this case the pipes & filters architecture is the right choice, because to switch between different evaluation modes we only need to replace a particular filter.

This component is quite important because the evaluation usually triggers the phase-switch from the Exploration phase to the Exploitation phase. A wrong evaluation in this case can lead to a too fast phase-switch which can affect classification performance.

3.2.6 Parameters Updater

To optimize data usage and obtain less biased data, could be useful to insert a Parameters evaluation step. At the moment the only affected parameters can be **sampling frequency** and **crawling depth**.

Sampling frequency optimization can lead to a better data utilization, both in storage and bandwidth. For example if a news source has a very low update frequency it is

useless to crawl it at every hour. It is better to crawl it once a day.

Figure 3.3: An example of *NBC News* update frequency

In Figure 3.3 is it clear that *NBC News* has a quite high update rate (2 news in a minute) and so do some chinese websites aswell.
In this case we'll need even to reduce the sampling frequency to avoid the risk of loosing information.

3.2.7 Extra considerations

Data needs to be stored, especially crawled data. This is due to the fact that it can be re-used to study more phenomena or, speaking of a production environment, to keep historic data and make further analysis on it.

We could also store the Typed Hypergraph generated by the two phases. It represents the actual knowledge of the system ed it has to be exploited at every time, even after a system shutdown.

Due to all these aspects, often, the just discussed components need to communicate with storage services.

For the sake of this work these were all local, but in a production environment they need to be as much available as possible and so need to be replicated.

This feature is nowadays offered by any Cloud Storage service, which can adapt to specific user needs.

Chapter 4

Experiments & Results

In this chapter we are going to examine some experiments and results. The discussion will be incremental. This is done to validate every statement we made in previous chapters. Every step we made studying this approach is motivated by some experiments and results made on real-world scenarios.

It was a try-and-act approach in which in first instance news sources were browsed by hand to find correspondences with the obtained results. This was a bit no more than a cold start. Over time the confidence with the news context growth and recognizing some behaviours became easier.

That's why, along with graphs and numbers, we'll provide evidences of real-world news sources, motivating those results.

4.1 Technological Set-up

We used two separate machines to study this approach and run experiments. They are such composed:

- **CPU**: Intel®Xeon®CPU E5-2620 v4 @ 2.10GHz (32 cores CPU)
- **RAM**: 64GB ECC-DIMM
- **Storage**: 1TB SSD storage
- **OS**: RedHat 7 CentOS

They worked pretty well but, as it is well-known, crawling is a very expensive operation, especially when made on multiple websites.

For this study we also needed a crawling over time, so we can easily say that one of these two machines was constantly working on crawling. The other one was used to effectively run experiments. These are no more than Spark Jobs with a lot (sometimes ~70 million tuples). This motivated the choice to dedicate a single machine to experiments.

We faced a bit of cold start due to the lack of data to work with; As we'll see further in this chapter we started working with 24 hours data, but that wasn't enough for the most part of websites. For the provided experiments we chose to work with a fixed sampling frequency of 1 hour in the first instance.
This choice was totally experience-guided and simpler to control.

4.2 Metrics

To evaluate a result we used three common metrics used in Information Retrieval[1]. We chose **Precision**, **Recall** and **F1-Measure** because they are well-known and it is quite simple to reason on them. We avoided accuracy due to the unbalancing of the involved classes. Given a news source, the amount of articles pages is a lot higher than sections and others pages. This would result in too optimistic results: if we classify everything as article the accuracy is still around 90%.

4.2.1 Precision

In Information Retrieval, **precision** is the fraction of retrieved documents that are relevant to the query. It quantifies, how many of the elements classified as relevant, are effectively relevant. For example, how many pages classified as articles are effectively articles.

$$precision = \frac{\{relevant\ documents\} \cap \{retrieved\ documents\}}{\{retrieved\ documents\}}$$

Is important to note that precision differs a lot from accuracy and the two metrics don't have to be confused.

4.2.2 Recall

In information Retrieval, **recall** is the fraction of the relevant documents that are successfully retrieved. It quantifies the ability of the system to correctly classify at least relevant documents.

$$recall = \frac{\{relevant\ documents\} \cap \{retrieved\ documents\}}{\{relevant\ documents\}}$$

In our study we aim to correctly spot sections. These are knowledge-entry points so a good recall score in recognizing sections is a good result because it represents the ability of our system to not loose information and knowledge.

[1]Information retrieval (IR) is the activity of obtaining information system resources relevant to an information need from a collection.[wik19b]

4.2.3 F1-Measure

F1-Measure (also called F1 Score) is a metrics that considers both Precision and Recall. It is defined as the harmonic average of these two metrics and it has a maximum value of 1, in which our prediction is 100% correct, and a minimum value of 0.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

It is used to briefly summarize precision and recall in a single metrics.

4.3 Stability experiments

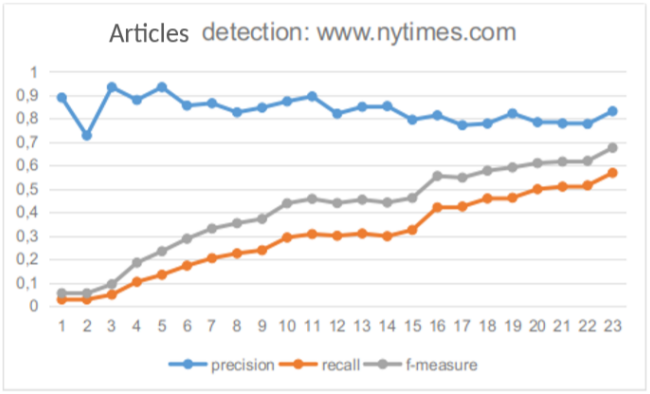
The first idea we gave in this work is that identifying temporal aspects and behaviours in pages can give us an idea of their nature. The first temporal feature we introduced (1.4) is **Stability**.

For our first experiments we randomly chose some news sources and observed them for 24 hours, because this way we would work with less data, making easier to reason and imagine real-world situations.

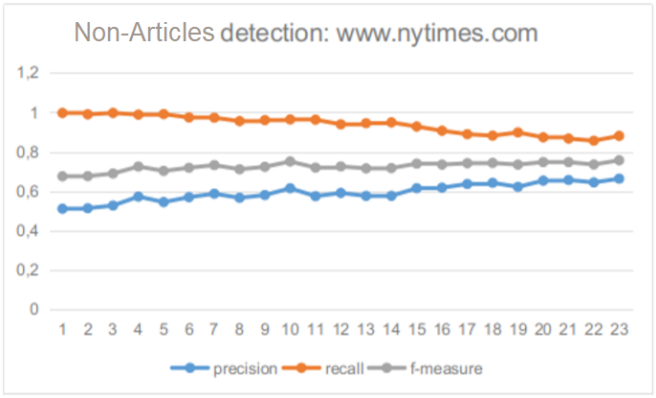
The site was crawled the at $depth = 2$ due to storage limitations. After we obtained the data we simulated, snapshot-by-snapshot, a process of *crawl*, *features-extraction*, *classification* and *model evaluation*.
At this point the system was trained to classify pages between articles and non-articles. This was due to the fact that Stability wasn't enough to identify other useful aspects.

For each news source we also handcrafted a golden set, in which was specified the right nature for every page. This operation was quite hard because of the size of some news sources.

4.3.1 New York Times: Stability



(a) Articles detection metrics in *www.nytimes.com*



(b) Non-articles detection metrics in *www.nytimes.com*

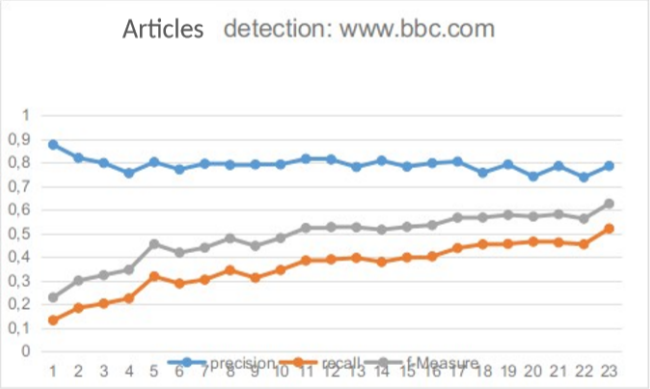
Figure 4.1: Comparison between articles and non-articles classification for *www.nytimes.com* on 24 hours

The previous Figure clearly shows that pages previously classified as Non-Articles, due to their initial high Stability value, are progressively reclassified in Articles.

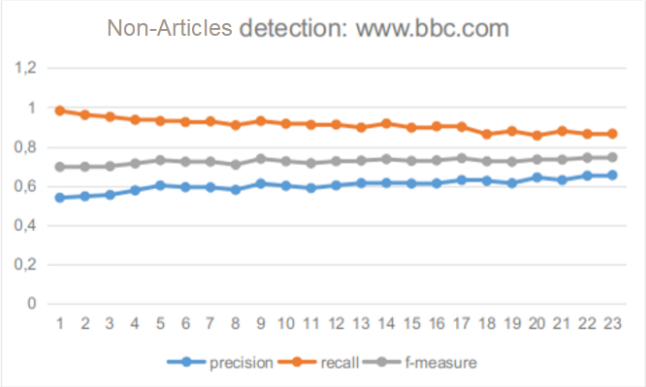
As we can see the recall of the Article detection grows over time while the precision is decreasing. Concerning Non-Articles, the precision of the Non-Articles detection grows, because the set it is refined snapshot-by-snapshot, while the recall decreases over time (but slowly).

This experiment showed to us that 24 hours of crawling are sometimes not enough to understand clearly the nature of pages. As we can see we have a monotone increasing trend for f1-measure in both the examples, but we cannot predict it in the future: it could stop increasing or maybe it could flatten in a plateau.

4.3.2 BBC: Stability



(a) Articles detection metrics in *www.bbc.com*



(b) Non-articles detection metrics in *www.bbc.com*

Figure 4.2: Comparison between articles and non-articles classification for *www.bbc.com* on 24 hours

The trend of the metrics showed for *BBC News* is the same seen in the previous experiment. In this particular case the growth of the article detection precision is slightly slower. This is due to numerous country-specific sections and articles belonging to

theese sections. All these distinct country-specific sections behave differently in terms of update frequency.

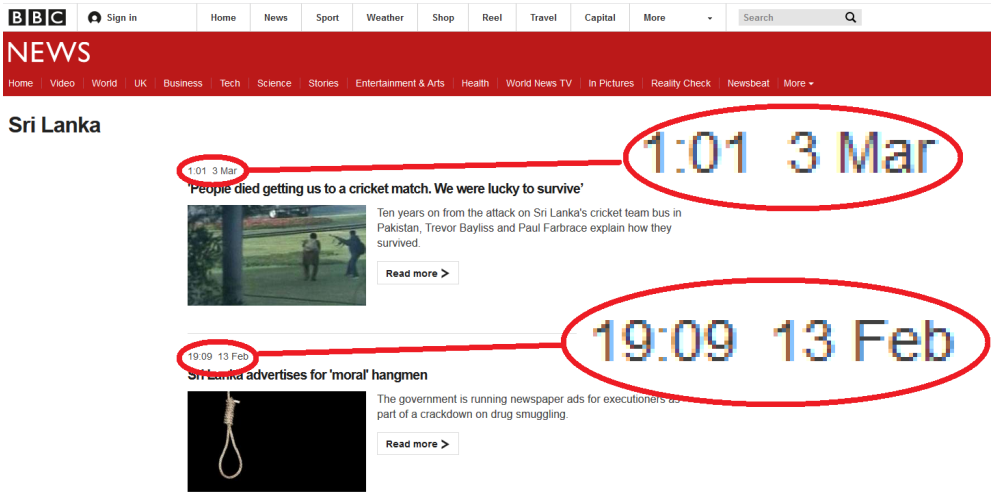


Figure 4.3: Sri Lanka section in *www.bbc.com*

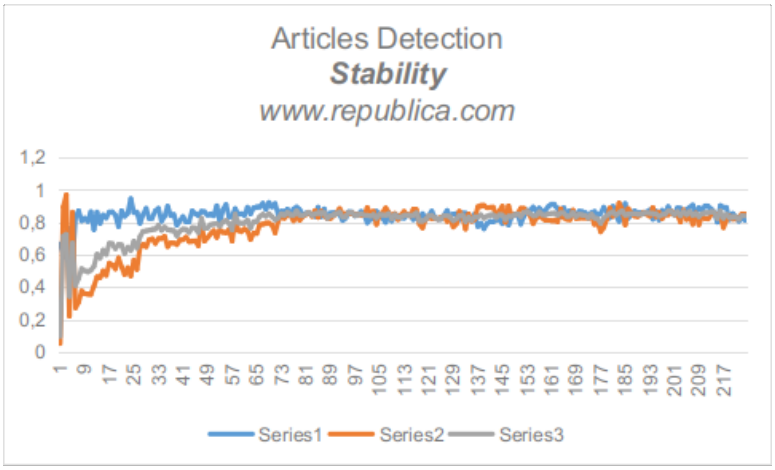
For example, let's think at *Sri Lanka-specific* section of *www.bbc.com*. As we can see in Figure 4.3, the update frequency (news over time) is quite low. In the provided image we have a 18 days delay between the highlighted articles.
This is not the case for a main section, in which we have 15-20 articles in a single day.

This is the motivation under the statement made in 1.4.3, in which we said that Stability, and in particular its regime reaching time, highly depend on the global dynamicity of the website.

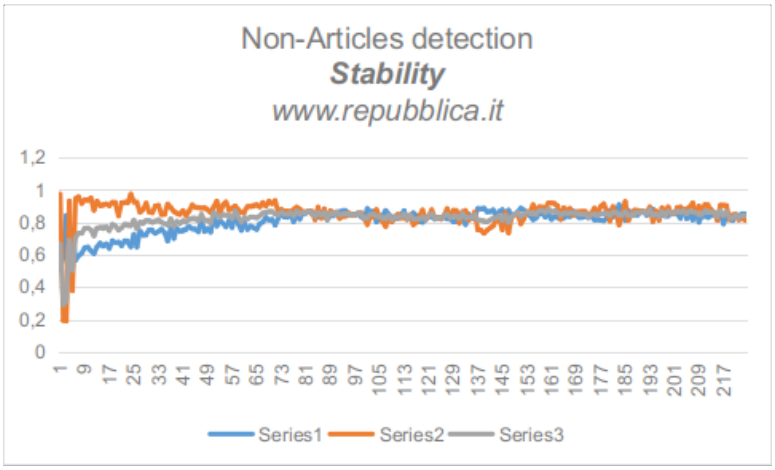
In this case *www.bbc.com* revealed itself as a quite low dynamic news source **overall**. With *"overall"* we meant including all the sections and country-specific sections.

4.3.3 Repubblica: a long tail example

Watching at 24 hours data we cannot understand quite well if the trend of the provided metrics will get better or will flatten at a certain point. This aspect was quite interesting so we decided to make an experiments a famous italian news-source: *www.repubblica.com*. *Repubblica* seems to be a quite simple news-source, with a good update frequency.



(a) Articles detection metrics in *www.repubblica.it*



(b) Non-articles detection metrics in *www.repubblica.it*

Figure 4.4: Comparison between articles and non-articles classification for *www.repubblica.it* on almost 10 days

As we can see in the Figure, F1-Measure for both the classes grows over time. It is also evident that it reaches its maximum around snapshot 80 and it remains stucked there

for the rest of the time.

This could be due to noise into the golden set (human error), but the cause could also be something wrong in the website.

During an hand-made exploration of the news source, we found a weird issue: some sections had something like 30-40 old articles at the bottom of the page. With *"old"* we mean two or three years old articles. These were not related to the ones above and are probably located there because they were the most clicked or viewed (at the time).

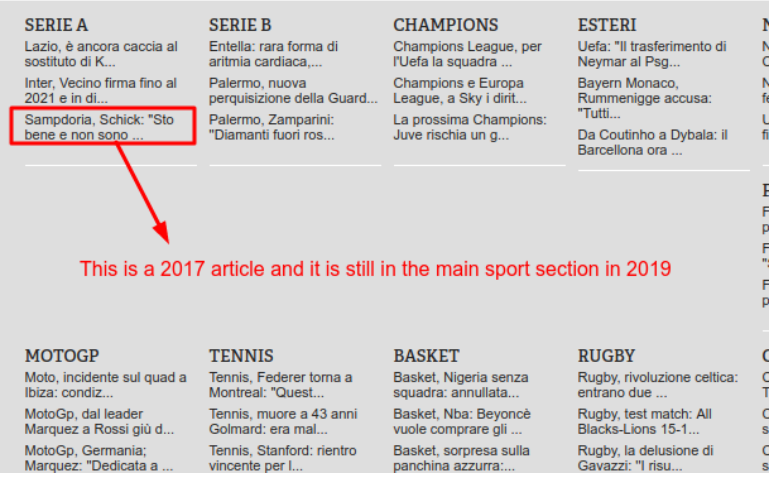


Figure 4.5: Example of a 2 years old footer article in *www.repubblica.it*

These are quite dangerous false positives because they are treated as Non-Articles and, later in the future, as Sections. The presence of these outliers cannot be avoided with provided approaches because, temporally speaking, they have a behavior very similar to sections: stable and no-position changes at all.
We can confirm that after months, those articles are still there.
We found something similar also in the spanish website *The Local* in which this situation was way more critical, even if we crawled up to 300 snapshots.

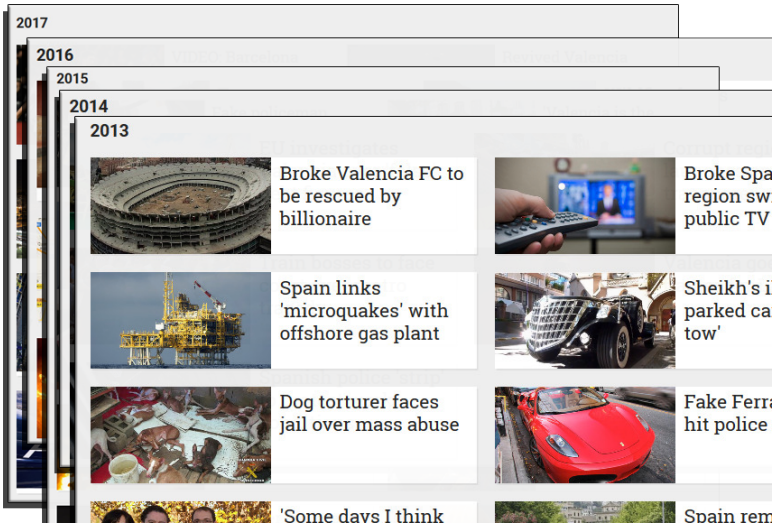


Figure 4.6: An example of *Articles Graveyards* in *The Local*

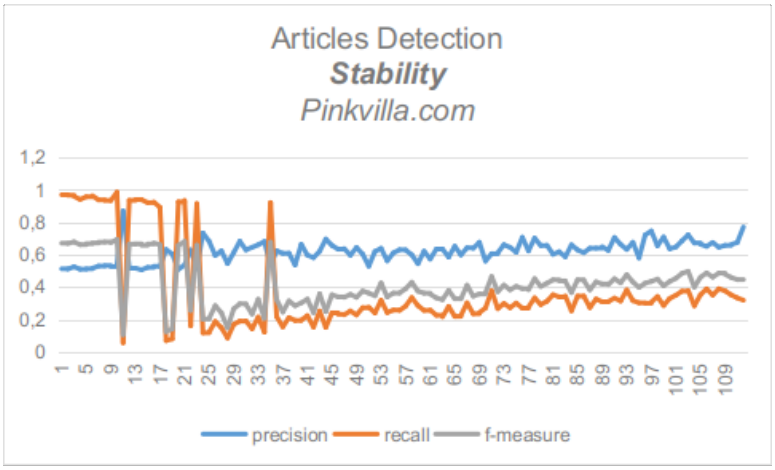
In this news source in every sections we can find up to six years old articles. This makes Stability completely useless, because the system is not able to catch any dynamic at all.

We'll focus more on this topic later in this work. Cases like these can be solved using **non-temporal approaches** and features (we didn't tested yet), like **Longest Non-Tag Sequence**. As we'll see in the last chapter, this feature simply counts the amount of non-tags words included in a tag and takes the highest one.

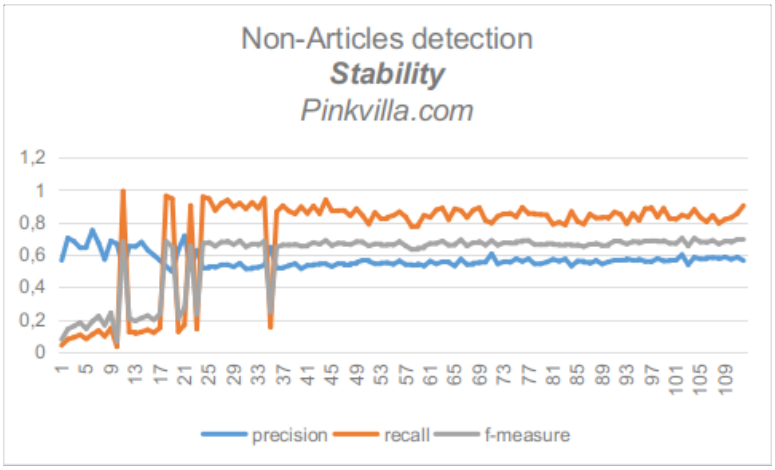
Articles and Others could have an high amount of consecutive text in a page (think to a privacy disclaimer). On the other hand, the longest non-tag sequence for a Section is nothing more than a snippet or incipit of a news.
This is clearly a limitation for this approach but it showed us that secluding only on temporal features was a bad idea.

4.3.4 Pinkvilla: considerations on section size

Pinkvilla is an indian gossip news source. It is interesting because it showed us an important aspect of Stability. This news source has sections with 40 articles per page. This is an enormous amount of articles for a section. Most news sources have around 10 to 15 articles per section.



(a) Articles detection metrics in *www.pinkvilla.com*



(b) Non-articles detection metrics in *www.pinkvilla.com*

Figure 4.7: Comparison between articles and non-articles classification for *www.pinkvilla.com* on almost 7 days

It is clear that having 40 articles in a section makes each of these articles to last more than a normal news source. As we previously said, sections are knowledge entry points

and what is inside them is still interesting and relevant, until they left them.

In this case relevance is quite biased by a weird design choice for the website. We'll see later in this work how the help of Hypertextual References Dynamicity can resolve this issue.

This showed that **Stability is also affected by the size of Sections** in terms of contained articles.

4.3.5 Other news sources: Stability

We made much more experiments trying to exploit Stability to catch some behaviours. We chose a baseline of 150 snapshots.

Figure 4.8: A chart showing many news sources compared by their articles classification ability using Stability

Figure 4.9: A chart showing many news sources compared by their non-articles classification ability using Stability

From the first Figure we can confirm that Stability heavily depends on the update frequency of the website and sections size. We had the best results in websites with a simple template and with a good update frequency.

This is due to the fact that websites with a complex template can create noise because an article could be part of it, resulting in a slightly biased data.
On the other hand, the second Figure shows quite low precision values, which can indicate what we just said. Some cases only need more time to get more precise.

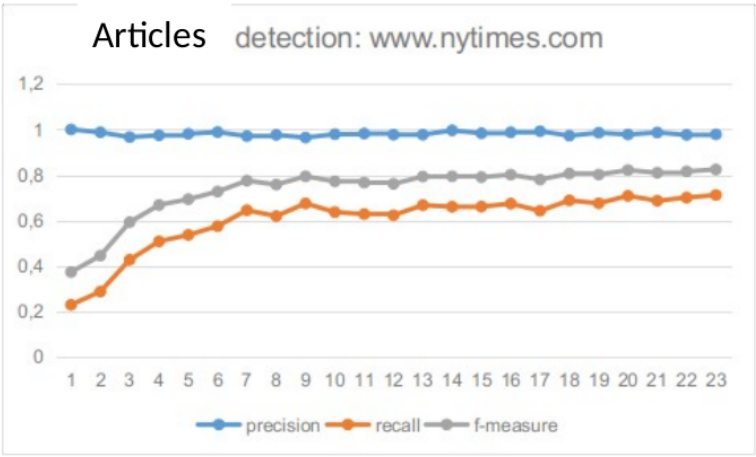
4.4 Hypertextual References Dynamicity experiments

Stability was not enough, we then moved to another feature, called Hypertextual References Dynamicity. We thought it could be interesting to study this feature alone, to study its real effectiveness. The setup of these experiments is the same seen in the

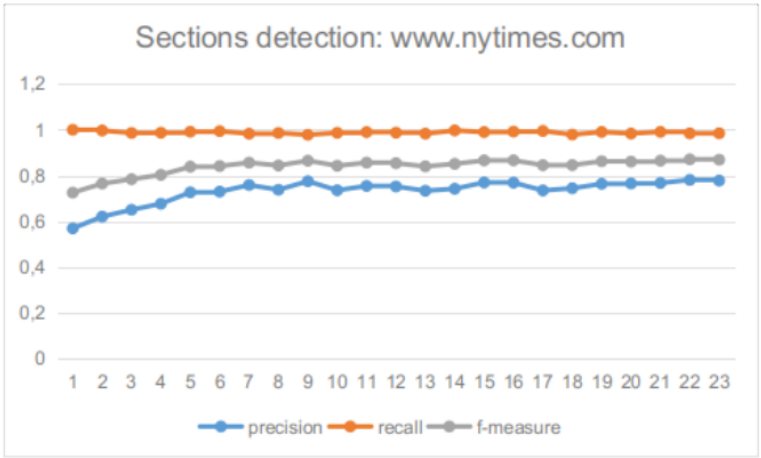
previous chapter:

- $depth = 2$
- fixed sampling frequency to 1 hour

4.4.1 New York Times: Hypertextual References Dynamicity



(a) Articles detection metrics in *www.nytimes.com*



(b) Non-articles detection metrics in *www.nytimes.com*

Figure 4.10: Comparison between articles and non-articles classification for *www.nytimes.com* on 24 hours using Hypertextual Reference Dynamicity

As we can see in 4.11 Articles detection accuracy quickly increases. this is due to the fact that updates are quite frequent on this website. This way our system can catch link occurrences movement very fast. Notice that we have slightly biased results because of crawling starting time. It happened an hour before lunchtime: exactly when many news sites update their site.

However the function seemed to reach a plateau probably due to the night hours, in which the updates are quite infrequent.

4.4.2 BBC: Hypertextual References Dynamicity

BBC gave us an important hint about this feature. This news source has sections in which articles link occurrences tend to appear and disappear.
These link occurrences don't move like other websites link occurrences do. They don't change position every snapshot-by-snapshot. Probably their presence into the section is guided by some sort of voting system (clicks or views).

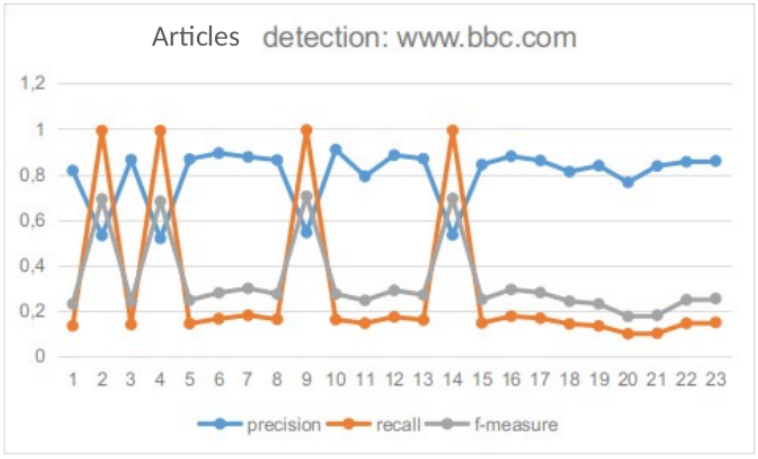


Figure 4.11: Articles detection metrics in *www.bbc.com*

Figure 4.12: Comparison between articles and non-articles classification for *www.bbc.com* on 24 hours using Hypertextual Reference Dynamicity

Articles detection appears to be pretty much random at this time. The system isn't able to identify Articles using Hypertextual References Dynamicity because it cannot catch movement of the link occurrences. The previously explained aspect which is causing this is evident in the portion provided in **??**. The link occurrences in the picture are clearly not sorted by their age. We have a 5 hours ago article, then a 7 minutes one and again a 2 hours ago one. That said we can state that Hypertextual References Dynamicity is affected by this kind of situations.

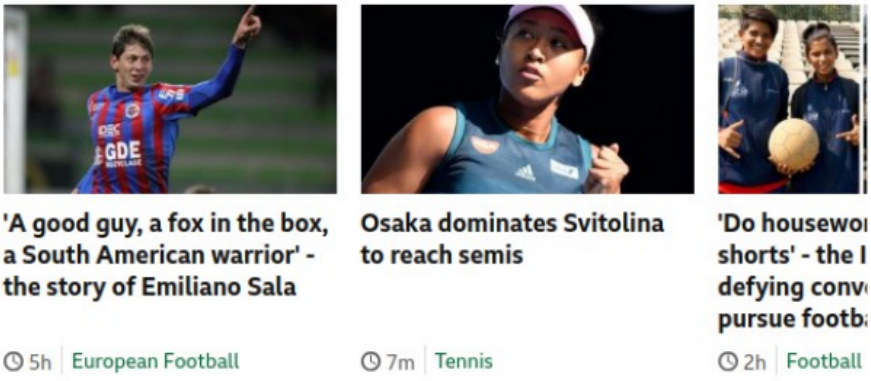


Figure 4.13: Badly temporized link occurrences in a section of *www.bbc.com*

4.4.3 The Irish Times: Hypertextual References Dynamicity

The Irish Times is quite simple news source. It has both good dynamic and update frequency.

Its sections are quite peculiar, with not so many articles. With our first experiments we noticed that something went wrong with Hypertextual References Dynamicity.

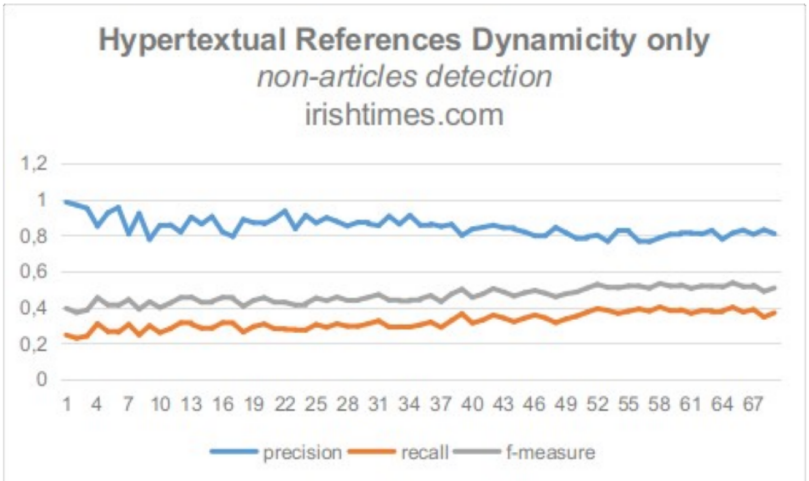


Figure 4.14: Non-Articles detection on *www.irishtimes.com* using Hypertextual References Dynamicity

As we can see, in this case Hypertextual Reference Dynamicity is misleading. This is due to a presentation issue. Every element of a link collection that contains a link to an Article, also contains a link to the corresponding Section. This results in sections having the exact same dynamicity as articles.

Figure 4.15: Issue with link occurrences in *www.irishtimes.com*

Notice that this is not a problem for Stability but only for Hypertextual References Dynamicity. This phenomena suggests us that Hypertextual References Dynamicity

could be affected by the way contents are presented. This is an aspect that an human cannot quite notice observing the page; he can notice a link occurrence of an article is moving but not a link occurrence referring to a section.

Fortunately our system caught this behaviour, allowing us to reason about it.

4.4.4 Other news sources: Hypertextual References Dynamicity

The experiment base for this batch of experiments is the same we discussed before. Here we are evaluating how good is Hypertextual References Dynamicity if used alone. We would like to built an overall idea about these features to better understand their relationships with the real-world.

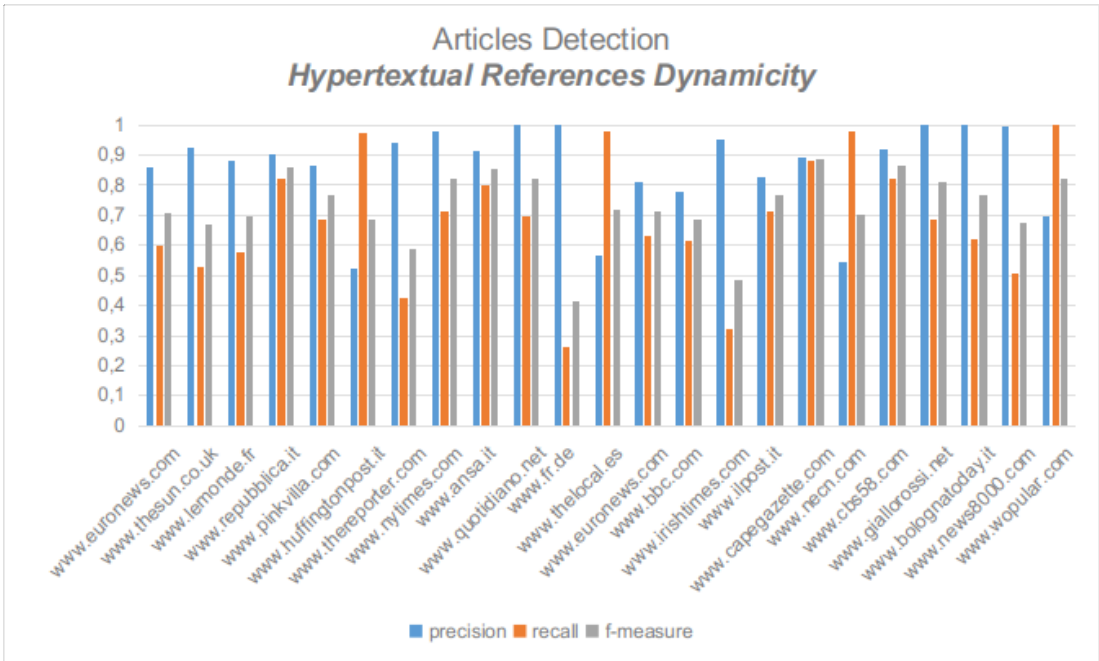


Figure 4.16: A chart showing many news sources compared by their articles classification ability using Hypertextual References Dynamicity

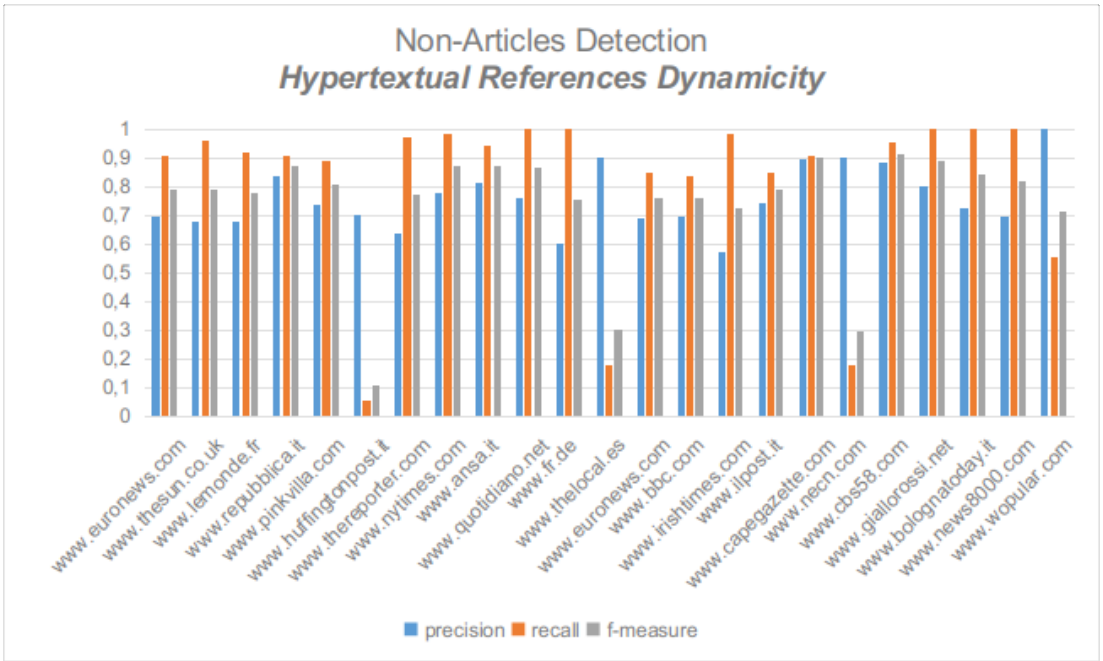


Figure 4.17: A chart showing many news sources compared by their non-articles classification ability using Hypertextual References Dynamicity

The results obtained using this feature alone reflect the fact that Hypertextual References Dynamicity is a less strong signal than Stability. It depends on different factors, such as the presentation of contents, update frequency and also the temporization of link occurrences.

A news source could have a weird way of representing articles and their link occurrences or could have a weird way to sort articles presented to the reader over time.

4.5 Multi-Feature Model experiments

In our multi-feature model we consider the possibility to use more features at time to determine the class of a page. As we'll see further in this section, even if we can provide and use every method or algorithm to achieve classification, the strength and solidity of our approach is obtained by the use of a voting model.

This is a concept borrowed from Ensemble Learning[2].

As said in 1.6, we follow the principle that the majority is right. this way we are resilient against misleading features.

4.5.1 Pinkvilla: Multi-Feature Model

As said in 4.3.4, *Pinkvilla* has some problems with Stability. This is due to the high amount of articles in sections, but also to the low updates frequency of some sections as we can see in 4.18.

Figure 4.18: Update frequency quite low for a section in *www.pinkvilla.com*

This two issues could be resolved using Hypertextual References Dynamicity. What we achieved is that using both Stability and Hypertextual References Dynamicity is a little bit better than using Hypertextual References Dynamicity alone, even if stability could be misleading in this case. This thanks to the voting model we previously spoke about.

[2] In statistics [...], ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. Unlike a statistical ensemble in statistical mechanics, which is usually infinite, a machine learning ensemble consists of only a concrete finite set of alternative models, but typically allows for much more flexible structure to exist among those alternatives.[wik19a]

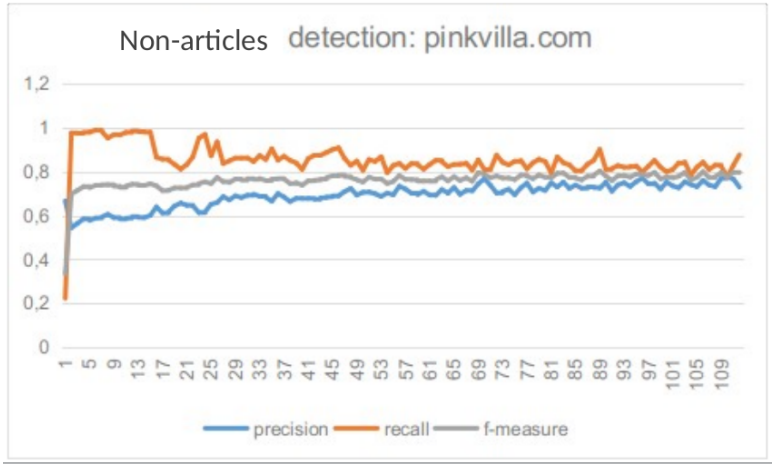


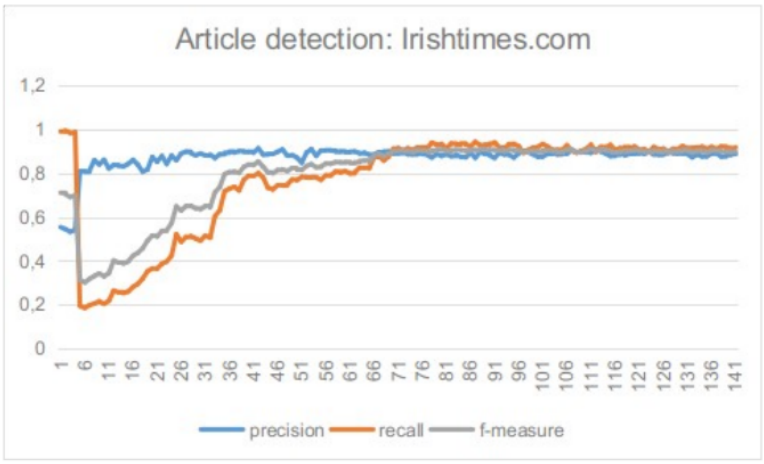
Figure 4.19: Multi-Feature model in non-articles detection in *www.pinkvilla.com*

Even after 110 snapshots, results are not so great but the trend of the goodness of the model is clearly growing. This is a clear signal that the website is not homogeneously dynamic. In this case, the system would need more snapshots to build a better idea of this news source.

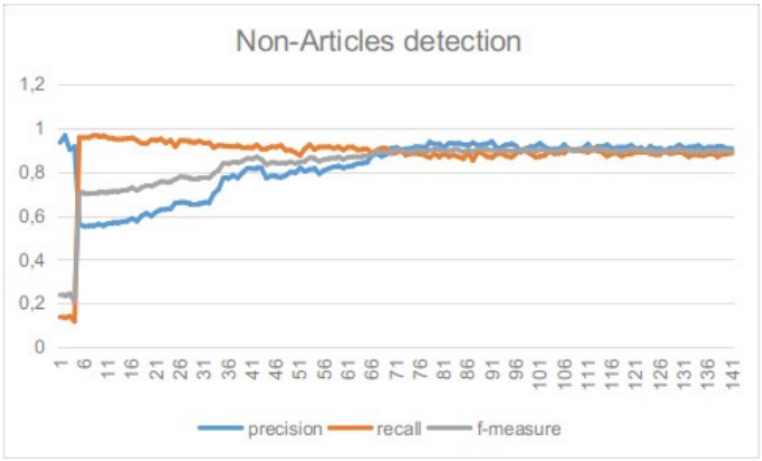
thanks to this experiment we stated that the multi-feature model along with is simple voting system makes the prediction more stable and less affected by misleading features.

4.5.2 The Irish Times: Multi-Feature Model Resiliency

The Irish Times suffers from the problem of the articles link occurrences bound to the link occurrence of the belonging section4.4.3. This led us understand that Hypertextual References Dynamicity was quite ineffective. using both Stability and Hypertextual References Dynamicity in this context makes our system to benefit from the good update frequency and well-sized sections of this news source.



(a) Articles detection metrics in *www.irishtimes.com*



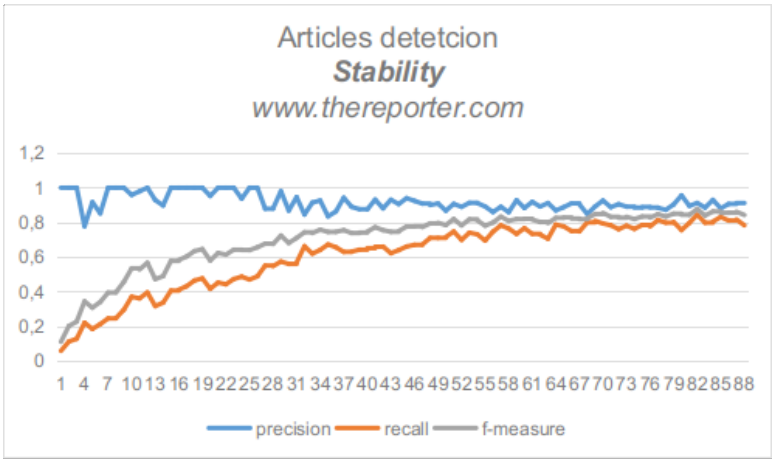
(b) Non-articles detection metrics in *www.irishitmes.com*

Figure 4.20: Comparison between articles and non-articles classification for *www.irishtimes.com* on 130 hours using Multi-Feature Model composed by St. and HRD

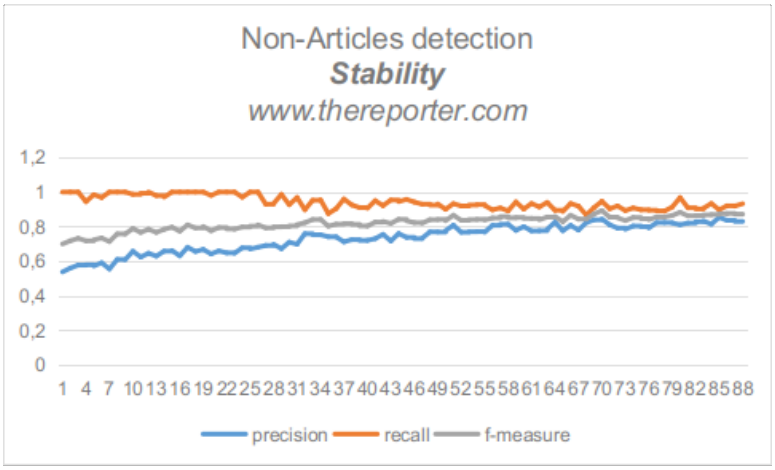
This is another great example about the resilience of this methodology: having a misleading feature (in this particular case Hypertextual References Dynamicity) doesn't negatively affect the prediction. For this news source Stability does a great job because old or no more relevant elements tend to disappear quite fast, letting our system to catch this behaviour.

4.5.3 The Reporter: Performance Boost

The previously analyzed cases are both unfortunate cases in which a feature was misleading. Using both features could also provide a performance boost to the systems, in quality terms. This is the case of *The Reporter*, a simple website with well-sized sections and good updates frequency (to be a local news source).



(a) Articles detection metrics in *www.thereporter.com*

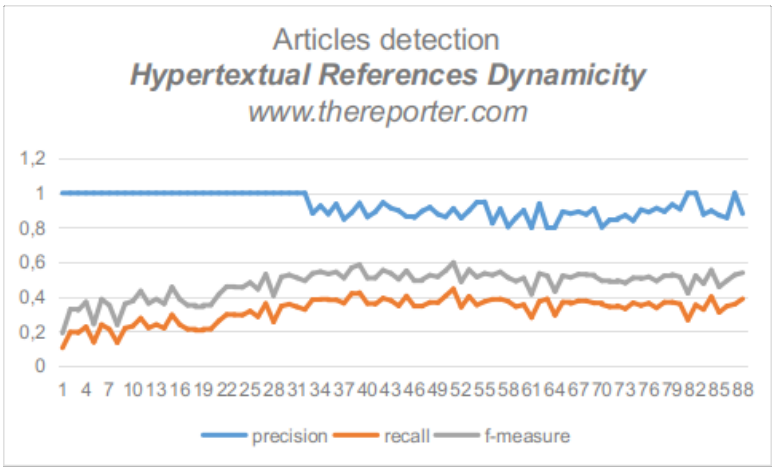


(b) Non-articles detection metrics in *www.thereporter.com*

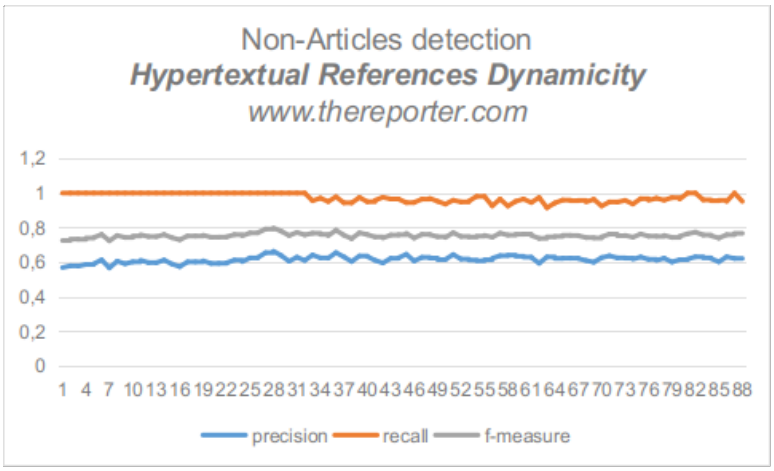
Figure 4.21: Comparison between articles and non-articles classification for *www.thereporter.com* on 3 days using Stability

As sections are well-sized, the appearing/disappearing rate is quite good so using Stability the system is able to distinguish between articles and non-articles. When

it comes to Hypertextual References Dynamicity things are not so good, because link occurrences movement are very few (being a local news website), so the system needs more time to exploit this feature the right way.



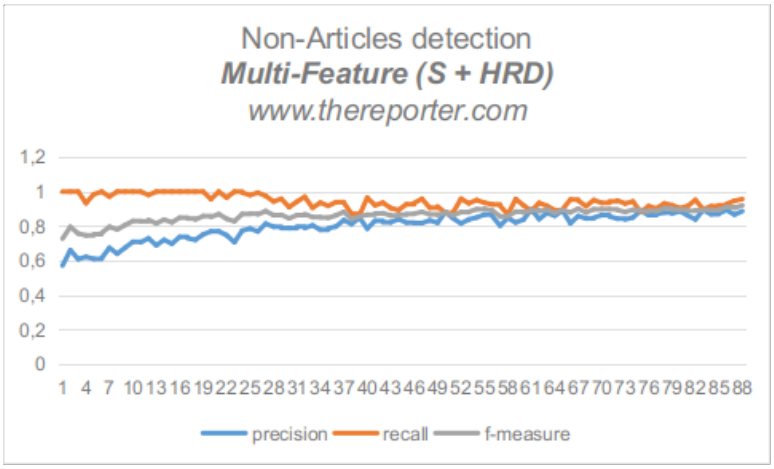
(a) Articles detection metrics in *www.thereporter.com*



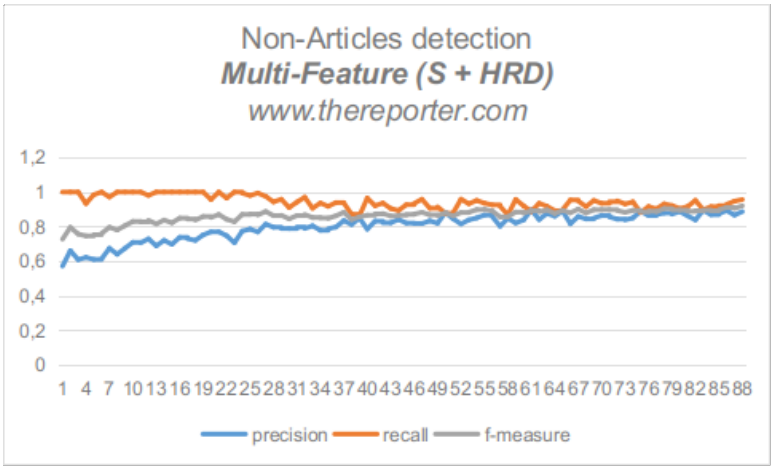
(b) Non-articles detection metrics in *www.thereporter.com*

Figure 4.22: Comparison between articles and non-articles classification for *www.thereporter.com* on 3 days using Hypertextual References Dynamicity

This result doesn't necessary mean Hypertextual references Dynamicity is not the right feature. For sure it is not the right one if used alone, because of its high regime reaching time due to the nature of the news source; but if it is used along Stability the obtained results are better than those obtained using only Stability.



(a) Articles detection metrics in *www.thereporter.com*



(b) Non-articles detection metrics in *www.thereporter.com*

Figure 4.23: Comparison between articles and non-articles classification for *www.thereporter.com* on 3 days using Multi-Feature Model (S + HRD)

This is probably due to the fact that if a link occurrences has some movement, but not so much, it can be confused with a steady one. To be sure that a small amount can identify an article behavior we can exploit Stability. If the movement amount is small, but still greater than zero, and if stability is quite low then it is an article.

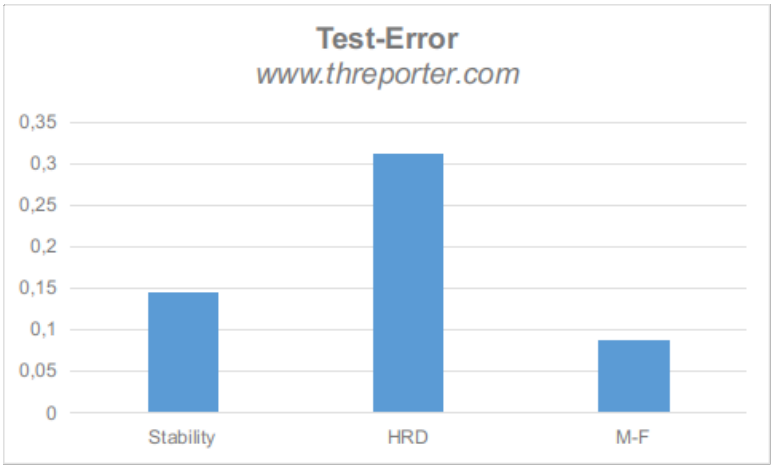


Figure 4.24: Test-Error lowest peaks for each model, calculated on *www.thereporter.com*

This demonstrates that using a Multi-Feature model can also increase performance compared to a single feature model. This is also motivated by 4.24, where we compared models on a test-error basis. **Test error** is calculated by predictions made on a never-seen set for the system. So, if we calculate the error on data which was unknown in the training phase, we are calculating the test error.

4.5.4 More news sources: Multi-Feature Model

Here we are going to provide the reader a more clear view over the adopted model. Multi-Feature model is used to guarantee strength to the prediction.

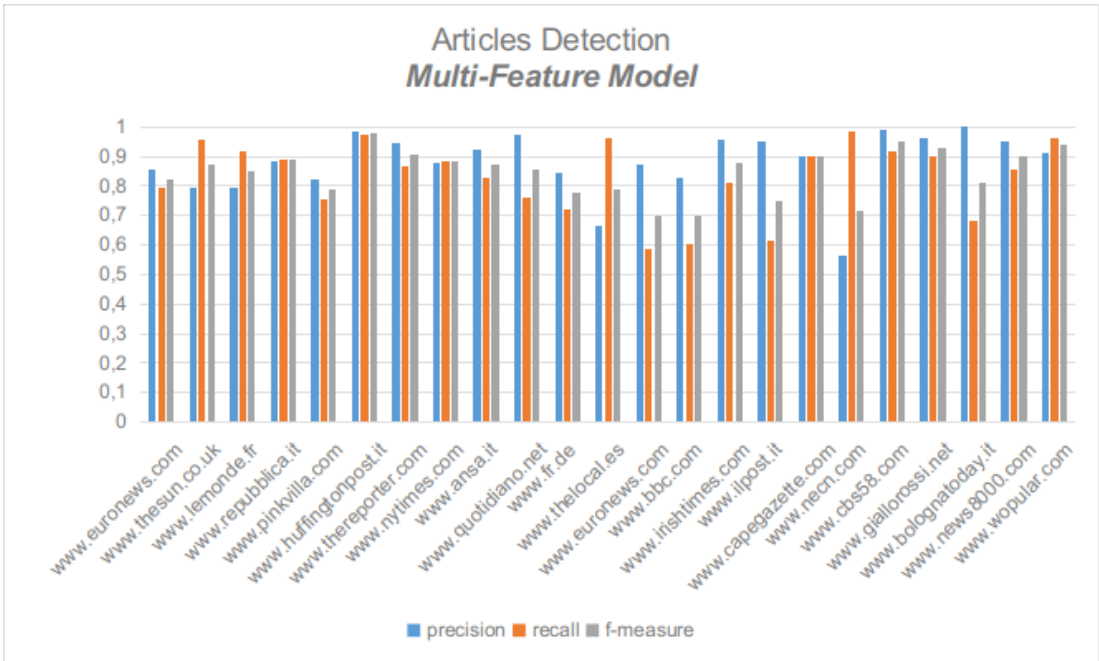


Figure 4.25: A chart showing many news sources compared by their articles classification ability using a Multi-Feature model

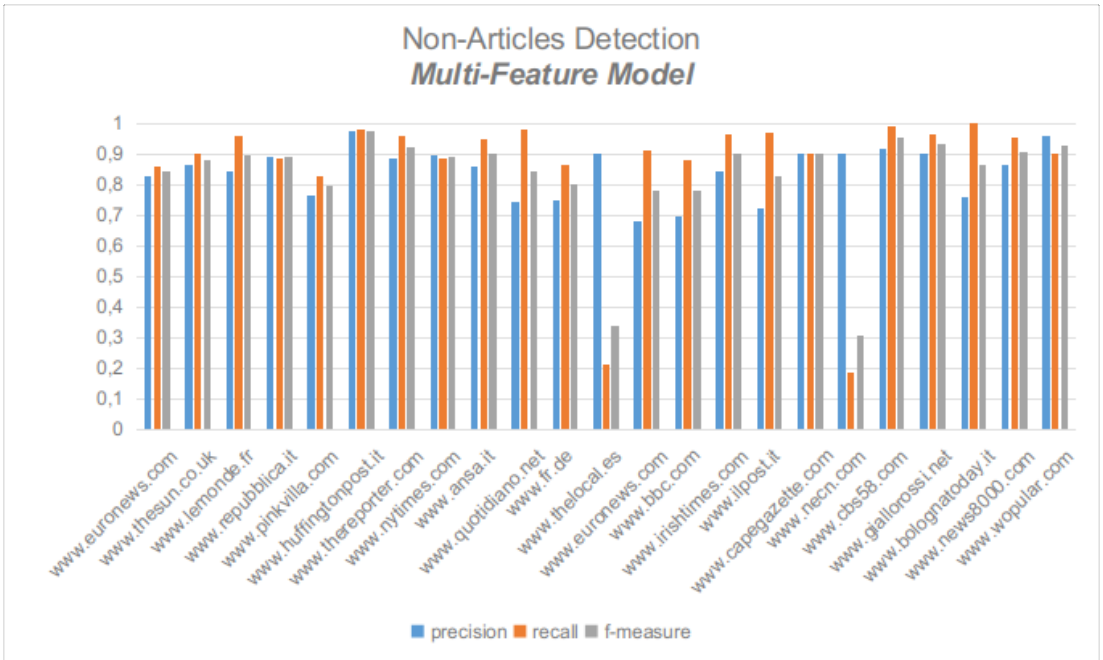


Figure 4.26: A chart showing many news sources compared by their non-articles classification ability using a Multi-Feature model

The results showed in the two Figures clearly reflect this aspect. The spikes are reduced because each news source can benefit from the features which are more evident for their case.
This is guaranteed by the previously discussed voting model

4.6 Section Driven Crawling

The last part of this chapter is made of experiments about Section Driven Crawling, we previously spoke about. These experiments are pretty much simulations of this process, having already crawled data.
Basically what we did was:

1. run an experiment about articles vs. non-articles classification
2. evaluate model for snapshot i
3. if the model is good enough go to the next step, else go back to 1
4. generate the typed hypergraph exploiting previous information
5. filter processed data on the hypergraph basis (sections only)
6. run experiment about articles v. sections vs. others for any other remaining snapshot

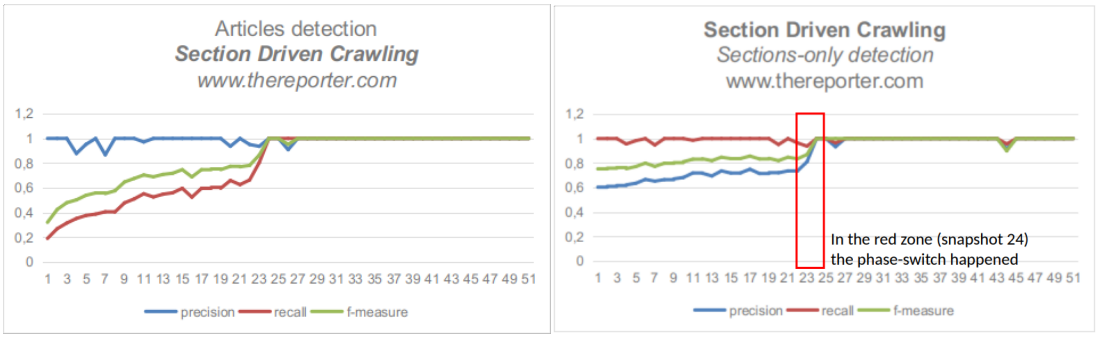
So, having already crawled data, we cannot test the parameters update. This is the only difference with the algorithm proposed in chapter 2.
As we said many times in this work, to study this approach and validate it we are currently using a supervised method to classify pages, so basically for both the phases (Exploration and Exploitation) we have hand-labeled data. In the next chapter we'll discuss some ideas for a totally unsupervised approach.
Having labeled data and using a supervised classification method it is also easy to

evaluate the produced model. This way we can choose a metric and a threshold for this one to trigger the phase-switch.

In our tests we chose f-measure on articles detection as phase-switch metric and we empirically chose a threshold of *0.86*. This choice was made because in the exploration phase, we give priority to articles, because we'll exploit this information in the next phase. When the system exceeds the provided threshold, the classification scope changes drastically from a two class (articles vs. non-articles) scope to a three class scope (articles vs. sections vs. others). This is the main reason we needed this algorithm. The system simply was not able to distinguish between these classes but now it is.

4.6.1 The Reporter: Section Driven Crawling

Our first experiments was *The Reporter*. We chose this news source due to its simplicity. We previously discussed system performance on this source in 4.5.3. It showed good results when using a Multi-Feature model made by Stability and Hypertextual References Dynamicity. However the model showed in 4.5.3 wasn't capable to distinguish between sections and others. As we can see in 4.23a, the model built a good understanding about articles. It started reaching the plateau in terms of quality around snapshot 24, more or less after a day.



(a) Articles detection metrics in *www.thereporter.com*

(b) Sections detection metrics in *www.thereporter.com*

Figure 4.27: Comparison between articles and section classification for *www.thereporter.com*

It is quite evident from the image that the phase-switch happens around snapshot 24. As we previously said this is the point were the system reached a good degree of knowledge about articles. That confirmed the need of a good phase-switch point to exploit previous knowledge.

From that point on, the classification is quite perfect both for articles and sections, but also for other, as we can see from Figure 4.28

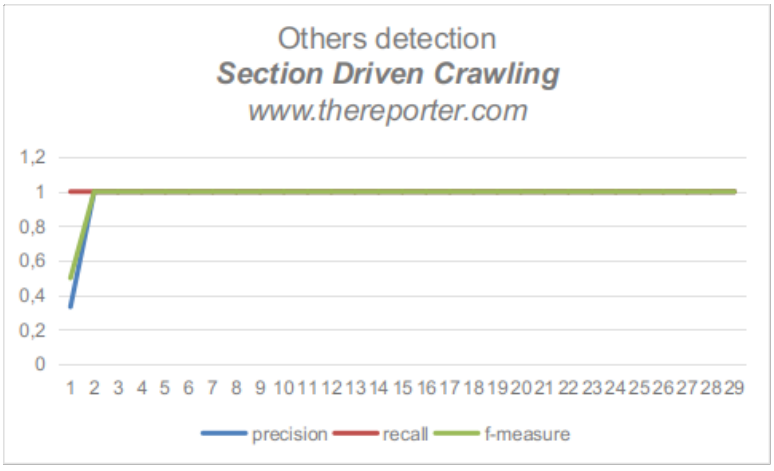


Figure 4.28: Others classification for *www.thereporter.com*

4.6.2 The Irish Times

Chapter 5

Related Works

In this chapter we will discuss the most interesting works regarding the field of automatic news extraction.
These can be divided into various categories. First of all, most of the proposed works are based on pre-obtained data. More generally, we can say that, in such works, gathering the data is not part of what is proposed.
We identified two major categories: **Data-Ready** and **Non-Data-Ready** (although these two categories will be treated at the end of this chapter because they include elements we are about to define).
The first one includes those methods that prevent the presence of initial data such as training pages, and the second those in which the gathering of data is an integral part of the proposed work.

According to the different mechanisms of extraction is possible instead a further classification in **Rule-Based extraction** and **Feature-Based Extraction**.

5.1 Rule-Based Extraction

This category includes all those jobs that aim to build or generate a rule-based Web Information Extraction Model (**WIE**). News sources, often respect presentation patterns guided by the specific domain.
For example, articles pages (always have an author, a title, a body and a date. One can model the structure of a page of a certain site, specifying the rules that the extractor must follow to find the actual content.

5.1.1 Hand-crafted models

This subcategory contains all those approaches in which, through human work, it is possible to create extraction models. For the news domain, this is a very safe approach because it allows to have really good results thanks to the fact that each model is manually calibrated for a specific news site.
This category includes systems like **WebOQL** [AM99] and **XWRAP** [LPH00].

WebOQL allows the user to manually build a very versatile model that adapts to different types of contexts (documents). It is based on a data structure called *HyperTree*. This is nothing more than a tree in which there are elements in the nodes. Most of the time these elements are structured. With different types of arches is it also possible to model references. Through an additional data structure the abstraction level is raised and it is also possible to model relationships between hypertrees. This way the user can query the system for some data.

XWRAP instead provides the user with an interface to build a custom wrapper used during the extraction phase. Although the interface can make you think of something intuitive, but technological details to know are not so few.

5.1.2 Conclusions

As already mentioned, these models are very precise but in a variable domain such as the one of news, a manual approach is unsustainable. It requires considerable knowledge and skills, even in programming (as in the case of WebOQL). It also requires the user who goes to build the model to deeply know the news source he is going to deal with.

5.1.3 Supervised Extraction

Instead of manually building models for a certain context, we can train a system that can extract characteristics from them, in a supervised way.
One approach could be to label a certain set of "training" data, providing a semantics to the elements that are part of it: a title, a body, an author, etc. in the case of news. This category includes works such as **DEByE**[LRNDS02] and **PNFS**[WXWD11].

The first work presents an approach to extract data from web sources, based on a small set of examples specified by the user. He specifies examples according to a structure of his choice. For the specification of these examples, the user can use a specific tool that adopts simple mechanisms, belonging to the visual paradigm. the use of this tool relieves him of a lot of technical knowledge. The examples provided by the user are then used to generate schemes that allow data to be extracted from new documents.

PNFS (Personalized News Filtering and Summarization on the web), on the other hand, is a system used for summarization and recommendation of news to the users. To be able to perform such operations, it has a phase in which it extracts news content and separates it from content that is not. The component that deals with the extraction was based on extraction rules based on the paths, in terms of tags, up to the content to be extracted.

5.1.4 Conclusions

This category requires less programming skills because it is "simply" a matter of annotating documents. Annotating an amount of data representing a set of all the possibilities is, however, quite complicated and burdensome. This is surely due to the amount of data, but to its generality as well. It is certainly a more efficient approach than the ones presented in subsection 5.1.1, especially when applied to the news domain. Said that we still have the disadvantage of knowing the domain and the particular site we want to do extraction on.

5.1.5 Unsupervised Extraction

This category bases its effectiveness on the principle that similar pages share similar structures and it could be very interesting to identify them. The main objective of these methodologies is to extract and find a common template or common features in the template of similar pages.
If you know which is the template of a news site, you can perform some kind of differential analysis and exclude the template from the content. This category includes **EXALG** [AGM03], **DEPTA**[ML16], **RoadRunner** [CMM$^+$01] ecc.

EXALG searches for template elements or common elements in a set of pages in order to isolate the content generated by a script and the content coming from a database. To understand this, it uses the concept of occurrence vector (tokens, which can be words or HTML tags). Elements with the same occurences vector belong to the same equivalence class. The assumption is that the elements of the page that are generated by the data and not by the template will not belong to any equivalence class or anyway to equivalence classes with characteristics below certain thresholds.

DEPTA, on the other hand, builds the tag-tree of a certain web page, then identifies regions where the data has the highest probability to b contained in. To achieve this it makes comparisons between adjacent tags in sub-trees. Finally, it combines subtrees with a certain degree of similarity. The assumption on which tags to analyze was valid a few years ago, but in most modern news sites it is no longer so easy and immediate.

5.1.6 Conclusions

These approaches are very effective (and efficient due to the fact they are unsupervised) but are also very fragile when something changes in the template. When working on a large number of pages, a small change in the template could result in a re-training of the model. Another crucial aspect is that even noticing such changes is quite difficult.

5.2 Feature-based Extraction

Recently, despite the fact that the value of the methods mentioned above is quite known, new methodologies are making their way using algorithms that search for and evaluate different features to support the task of extracting news. Instead of building an extraction model based on extraction rules, let's look for features (empirically recurring) in the pages in order to identify key points useful for the extraction. The methodologies that will be presented offer greater precision and more extensibility, which in the context of news sites is of vital importance. The problem often lies in the cost of the involved operations.

5.2.1 Vision-Based Extraction Models

The idea behind these methodologies is that the information of interest to the user is positioned and sized in a certain way within the page, and presented with a precise

style. This is done to make them even more usable. Tuning into this "channel" could lead to a clearer extraction of this data.

For example, in [ZSW07] an interesting methodology called **V-Wrapper** (or Visual-Wrapper) is proposed in which each element of the DOM-tree is a visual block. To understand how important a certain block is, different visual characteristics are evaluated on the same block. In news domain there are a lot of exploitable elements, such as positional and dimensional characteristics, as well as style (maybe the tags involved, or the font) and statistics (the number of hyperlinks, tables or images). The domain of our interest is often characterized by specific features, just think that the most important visual block of a page (the core of the news) could be the most central and larger than the others. Working also on a DOM view, it is also possible to understand the relationships between the various blocks. A "brotherhood" relationship (in the DOM-Tree) between two blocks of very different sizes might suggest a relationship between title and body, if maybe one of them includes some *heading* tags.

5.2.2 Block-Based Extraction Models

The idea behind these methodologies is always to identify blocks within HTML but to evaluate non-visual features. For example, **CETR**[WHH10] and **CEPR**[WLHW13] base their content search on features such as *tag ratio* and *path ratio*.
In [WH08] the relevant text of a web page is extracted by studying the *text-to-tag ratio* and instead in **MCSTD**[SLD+16] a page is preprocessed and then the *maximum continous sum of text density* is calculated. On the basis of this elements the extraction will be carried out. Methodologies of this type are more efficient than previous ones and are more resilient to changes. What needs to be assessed is whether the proposed assumption maintains its robustness for all sites and especially over time. This classification also includes [DLYD08] which proposes a generic model of news extraction based on

different heuristics such as the types of tags or the presence of some fixed points such as dates and relationships between information on the page.

5.2.3 Other features-based methodologies

In **TWCEM**[THF+18], an interesting work from 2018, a feature-based extraction model is proposed that revolves around the title of an article. The idea is that the title is a summary of the content and has a strong correlation with the body of the article. The title is extracted with a combination and comparison of tags (for example, imagine `<title>` with tags `<h>`).
Once you have associated a title to a web page you reprocess the content of this page to get the body: you clean the HTML from useless tags and search for blocks of content with more punctuation than the rest of the other blocks. This is because the block containing the body of an article often has more punctuation than other blocks. Once the candidate blocks have been identified an algorithm is used to find the *LCS* and among the blocks that exceed a certain threshold the one with the longest subsequence is chosen.

Other interesting works are [WCW+09] and [WHW+09] which always rely on combinations of features to extract the content of a news item. They are both visual features such as block sizes and positions, but also statistics such as word density and quantity.

5.3 Data-Ready Methodologies

The methods proposed in 5.1.1, 5.1.3 and 5.1.5 need to have available data build their knowledge and develop models. In such work it is assumed that these pages are already available, ready to perform the extraction. The task of obtaining these pages is therefore not the responsibility of the user. For the methods proposed in 5.2 the classification is looser because it is possible to use these technologies in a context *try-and-fail* so they

can be classified between this category and the next.

The approach proposed in this work aims to provide semantically labeled pages, which can therefore support methodologies which need data to work properly.

5.4 Non-Data-Ready Methodologies

These methods include extraction mechanisms that also include page retrieval functionality and represent, at present, our best competitors. At the moment there is only one system that is documented in the literature and that is able to achieve this: **News-Please**[HMBG17]. It is a generic, multi-language crawling and extraction tool that works *out-of-the-box* for a wide variety of sites. It allows you to extract publication dates, authors, news text, images, etc. All you need is the URL of the site you want to study. At the base of this very promising system there are several technologies and works that represent the current state of the art. As for crawling, a library for Python is used **scrapy**[scr18] and elements such as RSS feeds and sitemaps are examined where possible to understand what content to visit. As for the extraction, a high modularity is promised and what is included in the initial package are two extractors that represent the state of the art at this time: **Newspaper** and **Readability**.

Newspaper is an extractor specialized in articles. In its latest versions it also includes a feature for the download of articles from a news source if provided with a valid URL. This feature makes it in effect a crawling and extraction framework on a par with the aforementioned News-Please and a methodology for the section we are writing of. For the Focused-Crawling phase[1] uses very simple heuristics such as a similar-stopwording on URLs, going to mark as "not to visit" addresses that expose words such as *tags*,

[1] In Focuse crawling there is a Web Crawling of pages that satisfies a certain property. In this case "being an article".

career, *authors* and the like. Similar mechanisms are also used for content extraction, authors, dates, images, keywords and summaries. Most heuristics are based on empirically verified combinations of tags and attributes that seem to work well in many contexts.

Readability, on the other hand, is a library for Elixir, which is used only to extract content, author, title, dates and summarization. In [HMBG17] it is specified that its extraction characteristics are generally lower than Newspaper, but it does a better job in terms of extracting publication dates. Again, empirical heuristics are used with regard to HTML tags and their attributes.

News-Please also allows you to use combinations of extraction modules and at the moment this is the reason to present two extractors together in the basic version.

Conclusions & Future Works

This work proposed the study of a new approach to assess the nature of pages. This approach involves the use of temporal features. The target of this entire work is to understand the validity of this approach.
At the moment the system we built is oriented to the study and not to the production. This implies a series of limitations which will be discussed in this section along with some future solutions to each of these aspects.

The obtained results are pretty interesting. First of all we discovered that the temporal aspect is always present and visible by a machine and this was not obvious in the first iterations of the project. Asserting this aspect opens the way for a lot more work to do.

The results were, in some cases, quite good using a very small subset of features. On the contrary a system able to scale on a very large number of news sources can't rely on a set like this. Perhaps we'll need further features along with further experiments.
Another aspect we are going to considers is the automation and un-supervision. We ascertained that there are observable characteristics that can guide a machine in the choice of the nature of a page, but since this is a study we couldn't use a totally unsupervised approach from the beginning.
Once we stated that certain features are working well, we'll for sure move towards a

totally unsupervised solution.

Also the scope of the crawling deserves some attention.

Comprehensible Interface

At the moment the system is pretty difficult to use. Only who have built it can actually use it quite easily. Since this work cannot involve a single person, but probably needs a small team of co-workers, developing a better interface would be the first step.

This step is very important, because the study did not end with this work. Studying phenomena encountering technological limitation such the interface of the tool can really slow down the project or led to wrong results.

That's why we are currently studying the best (which is even the simplest for the moment) way to implement an interface for this tool. This interface should allow the user

- to easily extend the interface with every new feature or approach the team wants to study
- to recognize every type of experiment using it
- to select some parameters

Only this way the study can be flawless letting the team to work properly on the very problem and not only on technological difficulties.

Unsupervised Approach

As we previously said the proposed automation is just for the moment only theoretical, because all the methodologies we used needed human work.
This choice had very strict motivations. First of all without supervision we couldn't effectively quantifies our results. The presence of hand labeled golden set allowed us both to train the system but also to verify the goodness of the model.
News domain is such big that is not possible to check by hand if the results are good, so we needed an automatic method to do it. In fact, in the future we we'll move towards a totally unsupervised approach, but only when a reliable set of features has been found.

Challenges

Results obtained with a supervised approach can be a baseline for the study. There is also another motivation under the choice to start with them (and the simplest ones around). We are aware that, in first instance, results obtained with an unsupervised approach could be drastically worst than the baseline. But it is also true that this is a domain-specific work so we have a lot of information to exploit.

Solutions

As we previously said, to classify pages we trained decision trees and let them voting on the right class. This approach could still be used, also in an unsupervised approach. A very interesting article is proposed in [Vor]. In this article the idea is to exploit an *Unsupervised Decision Tree*. Basically to train a decision tree (actually in the random forest suit) a labeled data is needed. This is achieved by a simple clustering algorithm like *K-NN* or *K-Means* where the labels are the clusters. It is obvious that the clustering/labeling algorithm could be whatever the user wants; it could also be a proprietary algorithm specific for the news domain. After this automatic labelling phase we could

train decision trees as we did for this study. Decision Trees provide many advantages. They are pretty simple to understand and it is easy to reason on them, they haven't parameters to set and they are not affected by the unbalancing of classes. This last aspect is crucial for the news domain due to the fact that articles are much more than sections and others.

System Parameters

The proposed prototype relies on a few parameters. We can distinguish them in two different dimensions.

They can be different in the way they are provided. They can be provided statically, in the first run of the application or they can be adjusted dynamically during the execution.

They can have a different grain: some are system-specific and some are source-specific. It it clear that is impossible to tune source-specific parameters if the target is to scale on a large number of sources. So we have to move to a solution in which the majority (or even all) the parameters, such *sampling frequency* and *crawling depth* are dynamically adjusted.

This aspect in currently under study and implementation and it is base simply on the average stability of the entire news source and on the freshness of the link occurrences gathered in the current snapshot.

Crawling Scope

At the moment the scope of the crawling phase is news-source oriented. For a much finer and optimized crawling, changing the scope of it would be a quite good step.

Aiming to a news-source-section-oriented approach, instead, allows the system to estimate models and parameters on a section basis and no more on the entire website.

This will result in a more precise crawling. For example we'd have two different sampling frequencies for *London Local News* and*Sri Lanka local News* in the same website.

At the moment, this aspect is under study and testing because we think that is crucial for the purpose of the system we want to build.

More features

As previously said, our feature set is quite simple, maybe too much. There are more features we could exploit, some of them are temporal, and some are not.
It is very important to say that non-temporal features could be very useful in this case. We studied many other features than the presented ones in this work. Some of these were temporal and some were non-temporal.
Their calculation was not perfect and there were some cases in which they didn't work quite well. They were very interesting and they deserve more study.

Longest Non-Tag Sequence: A Non-Temporal Feature

A way to get if a page is a section or not is considering if it is has very long text sequences or not. This is more a characteristic of articles or others, where a very large text block is the main content of a page.
This was quite easy to implement and exploit, and indeed in some phases of the study it gave a boost to the performance. However, it is not interesting in temporal terms, so we decided to not cover it in the main work. The goodness of the results obtained exploiting this feature, suggested us that we should not seclude only on temporal features, considering that in this way we cannot cover all the news sources, for example due to the *Articles Graveyards Problem*.

Articles Graveyards Problem

During this study we encountered many news sources showing a weird phenomena. They had dead sections or (part of sections) reachable for a quite high depth, like 1 or 2. These sections have 2, 3 or more, years old articles inside. We started to call these kind of sections *Articles Graveyards*.

We can imagine a news archive as the simplest articles graveyard. When an article is too old to be showed in main sections it is moved to this archive. Temporally-speaking articles inside these graveyards are *dead* (this is the motivation under the name of the phenomena). This means that we cannot catch any temporal behaviour about them, so our approach would be completely useless for the most part.

The elements in these graveyards are very stable, due to its ease of reachability and also steady because they have no reason to move in the page. This would led the system to think we are in front of a very large number of sections.
The only way to understand the real nature of these pages is to exploit non-temporal features along with previously built knowledge. Non-temporal features like the one proposed above or the ones proposed in the previous chapter, can be effective in a context in which we have no hope to understand the nature with a temporal approach.
This problem is currently under study because we think that can be an obstacle for the utilization of the proposed approach.

Referring Page Changes Amount

As sections are quite stable and steady elements they are not so likely to change positions. There are instead some pages (link occurrences) which migrate to other pages over time. Let's imagine a website which has both common sections and topic-specific sections. In this case it could easily happen that an article is published on the topic-

section first and then into the more general one. This is a behaviour that only articles have and it is exploitable. We can imagine that once we generated the Hypergraph typed or non-typed we could catch some pages moving around. These pages are the articles.
We already studied this phenomena but the technological implementation of the quantification of this behaviour was far from perfect. However it looks like this is a quite good signal because it is able (Like stability) to clearly identify a single class. This feature could be added to the starting feature set and used in the Exploration phase of the algorithm.

Ingoing Links Amount Over Time

Sections are always part of the template, so they are likely to be reachable from every single page of the website. This means that an outgoing link to a section is present in every page. One we build the non-typed hypergraph we can exploit this an calculate how many ingoing links a page has. If we do this overtime we'll see that the amount of ingoing links for a section will always increase while at a certain point (sometimes depending on the update frequency) aritcles one will stop, because it became too old and starts to disappear from pages.

Microservices Architecture

To scale on an high amount of news source we also need an architecture that can support this characteristics. With the advent of Cloud technologies, high load computational infrastructure become within's everyone reach.
Microservices Architecture's has the benefit of decomposing an application into different smaller services is that it improves modularity. This makes the application easier to understand, develop, test, and become more resilient to architecture erosion. It parallelizes development by enabling small autonomous teams to develop, deploy and scale

their respective services independently.[wik19c].

Scaling is an essential feature for a system involved in the news domain due to the crawling power but also because the amount of resources to extract features and train the system on an high amount of news sources is quite huge.
We are thinking to move on such architecture in the future, once the systems characteristics would be more clear, defined and stable.

Bibliography

[AGM03] Arvind Arasu and Hector Garcia-Molina. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348. ACM, 2003.

[AM99] Gustavo O Arocena and Alberto O Mendelzon. Weboql: Restructuring documents, databases, and webs. *Theory and Practice of Object Systems*, 5(3):127–141, 1999.

[CMM+01] Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo, et al. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118, 2001.

[DLYD08] Yongquan Dong, Qingzhong Li, Zhongmin Yan, and Yanhui Ding. A generic web news extraction approach. In *Information and Automation, 2008. ICIA 2008. International Conference on*, pages 179–183. IEEE, 2008.

[HMBG17] Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. news-please: A generic news crawler and extractor. In *15th International Symposium of Information Science (ISI 2017)*, pages 218–223, 2017.

[LPH00] Ling Liu, Calton Pu, and Wei Han. Xwrap: An xml-enabled wrapper construction system for web information sources. In *Data Engineering,*

2000. Proceedings. 16th International Conference on, pages 611–621. IEEE, 2000.

[LRNDS02] Alberto HF Laender, Berthier Ribeiro-Neto, and Altigran S Da Silva. Debye–data extraction by example. *Data & Knowledge Engineering*, 40(2):121–154, 2002.

[ML16] A. Manjaramkar and R. L. Lokhande. Depta: An efficient technique for web data extraction and alignment. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2307–2310, Sep. 2016.

[scr18] Scrapy documentation. `https://docs.scrapy.org/en/latest/`, 2018.

[SLD+16] Kai Sun, Miao Li, Jinhua Du, Lei Chen, Zhengxin Yang, Yi Gao, and Sha Fu. Web content extraction based on maximum continuous sum of text density. In *Asian Language Processing (IALP), 2016 International Conference on*, pages 288–292. IEEE, 2016.

[THF+18] Zhen Tan, Chunhui He, Yang Fang, Bin Ge, and Weidong Xiao. Title-based extraction of news contents for text mining. *IEEE Access*, 6:64085–64095, 2018.

[Vor] William Vorhies. Have you heard about unsupervised decision trees.

[WCW+09] Junfeng Wang, Chun Chen, Can Wang, Jian Pei, Jiajun Bu, Ziyu Guan, and Wei Vivian Zhang. Can we learn a template-independent wrapper for news article extraction from a single training site? In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1345–1354. ACM, 2009.

[WH08] Tim Weninger and William H Hsu. Text extraction from the web via text-to-tag ratio. In *Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on*, pages 23–28. IEEE, 2008.

[WHH10] Tim Weninger, William H Hsu, and Jiawei Han. Cetr: content extraction via tag ratios. In *Proceedings of the 19th international conference on World wide web*, pages 971–980. ACM, 2010.

[WHW+09] Junfeng Wang, Xiaofei He, Can Wang, Jian Pei, Jiajun Bu, Chun Chen, Ziyu Guan, and Gang Lu. News article extraction with template-independent wrapper. In *Proceedings of the 18th international conference on World wide web*, pages 1085–1086. ACM, 2009.

[wik18] Wrapper (data mining), May 2018.

[wik19a] Ensemble learning, Feb 2019.

[wik19b] Information retrieval, Jan 2019.

[wik19c] Microservices, Feb 2019.

[wik19d] Silhouette (clustering), Jan 2019.

[WLHW13] Gongqing Wu, Li Li, Xuegang Hu, and Xindong Wu. Web news extraction via path ratios. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2059–2068. ACM, 2013.

[WXWD11] Xindong Wu, Fei Xie, Gongqing Wu, and Wei Ding. Personalized news filtering and summarization on the web. In *2011 23rd IEEE International Conference on Tools with Artificial Intelligence*, pages 414–421. IEEE, 2011.

[ZSW07] Shuyi Zheng, Ruihua Song, and Ji-Rong Wen. Template-independent news extraction based on visual consistency. In *AAAI*, volume 7, pages 1507–1513, 2007.